A SIMPLIFIED BIM DATA REPRESENTATION USING A RELATIONAL DATABASE SCHEMA FOR AN EFFICIENT RULE CHECKING SYSTEM AND ITS ASSOCIATED RULE CHECKING LANGUAGE

A Dissertation
Presented to
The Academic Faculty

by

Wawan Solihin

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Architecture

Georgia Institute of Technology
May 2016

A SIMPLIFIED BIM DATA REPRESENTATION USING A RELATIONAL DATABASE SCHEMA FOR AN EFFICIENT RULE CHECKING SYSTEM AND ITS ASSOCIATED RULE CHECKING LANGUAGE

Approved by:

Dr. Charles M. Eastman, Advisor
College of Architecture
Georgia Institute of Technology

Dr. Thomas Liebich
Director,
AEC3 Deutschland GmbH

Dr. Russel Gentry
College of Architecture
Georgia Institute of Technology

Dr. Robin Drogemuller
School of Design Office,
Complex Urban Systems Design
Queensland University of Technology

Dr. Godfried Augenbroe
College of Architecture
Georgia Institute of Technology
Chapter

Dr. Robert Amor
Department of Computer Science
The University of Auckland

Date Approved: November 18, 2015

To my dearest wife Lili for her loving and unconditional support, my children Grace, Sebastian and Timothy for their understanding, Chuck Eastman for his great guidance and support,

.. and to my Dear Heavenly Father who has allowed me to complete this lifelong dream.

ACKNOWLEDGEMENTS

Completing this dissertation is really an accomplishment of my lifelong dream. It would not have been possible without many different parties who shaped almost every aspect of this research, from a decision to take up the research late in my life, to the possibility of completing the study in an extremely quick fashion with just a little more than two years. All this would not have been possible if not because of the Lord's invisible hands that brought into my life many people and different experiences over the past 20 odd years that equipped me to achieve this. He allowed me to get to know my advisor, Prof. Chuck Eastman, who graciously and enthusiastically supported me in this research work. His role as a mentor, an advisor and a partner in exchanging ideas has tremendously helped me with the direction that I needed to take for this research.

I thank Prof. Fried Augenbro and Prof. Russel Gentry who are in my thesis committee for their enthusiastic support and guidance as well. In addition I would like to thank Prof. Robin Drogemuller, Queensland University of Technology, who led a code checking project in 2004 in Australia, Dr. Thomas Liebich, who is the leader in the IFC model development, and Prof. Robert Amor, who is the Head of Department (Computer Science) of the University of Auckland. They agreed to be members in my thesis committee despite their busy schedules and endured reviewing this lengthy thesis. I also thank Prof. John Peponis who was the Head of the PhD program then. His assistance in getting all the formalities and arrangements for my admission and other administrative matters I greatly appreciate. There are also others who have given me inspiration, challenges and guidance in various ways: Prof. Dirk Schaefer, who is now with the

University of Bath, Prof Leon F McGinnis, Prof. Ghang Lee, Yonsei University, Prof. Jinkook Lee, Hanyang University, Prof. Rafael Sacks, Technion University.

I was very privileged also to get to know several friends in the Digital Building Lab and fellow classmates who have made my stay in Georgia Tech less stressful and a memorable one: Yong-Cheol Lee, Donghoon Yang, Kereshmeh Afsari, Mehdi Nourbakhsh, Samaneh Zolfagharian, Shani Sharif, Sang-Pil Lee, Lisa Lim, and Marisabel Marratt.

My gratitude also goes to Anand Mecheri, the CEO of Invicara, who provided me with moral support and flexibility in term of my engagement with the company, without which the pursuit of this research would have been much more difficult to do. Autodesk also played an unlikely role helping me in my research indirectly, from the work experience I had with the company to other special circumstances that gave me an opportunity to decide to pursue this research. Also to Raymond Tan, CEO of novaCITYNETS, where I worked on CORENET ePlanCheck for many years. The challenges encountered during the development of ePlanCheck motivated me to work on this topic for my research even after many years had passed.

My family members are simply wonderful people I have in my life whom the Lord has bestowed upon me, especially my wife Lili. It was through her support and encouragement that I took the big decision to take a break from my regular employment and to spend the next 2.5 years slogging day and night over my studies. Her patience and her dedication to take care of the rest in the home especially when I was away are immeasurable. Her sweet smiles provided much needed energy to sustain the hard work throughout these years. Also my children Grace, Sebastian and Timothy who have had to

endure my absence during the course of my studies. Grace even helped me in proofreading the thesis. These are all truly great blessings that I enjoy.

Best of all, it is the Lord my living God who allows me to do all these. It is He who led me to learn various skills and to obtain experience in various fields that turned out to be so essential for this research. This thesis is essentially the culmination of knowledge and experience in the RDBMS, GIS, building code checking, programming language, geometry and graphics, etc. that I accumulated in the past 20 odd years by the grace of God. I am always amazed at how His hands were in this so long before I even realized it. I can only give Him thanks with all my heart and give Him all the glory that He deserves.

TABLE OF CONTENTS

LIST OF TABLES

Page

LIST OF FIGURES

Page

LIST OF SYMBOLS AND ABBREVIATIONS

2D	2 Dimensional, refers to a coordinate system that is made of two axes, i.e. plane.
3D	3 Dimensional, refers to a coordinate system that is made of three axes, i.e. volume.
AABB	Axis Aligned Bounding Box
AEC	Architecture, Engineering, and Construction
ANTLR	Another Tool for Language Recognition. It is a parser generator tool for reading, processing, executing, or translating structured text or binary files. It is an open source project that helps to generate a parser that can build and walk parse trees (http://antlr.org). The prototype work in this research uses ANTLR for parsing BIMRL. It uses ANTRL version 4.
API	Application Programming Interface
BCA	Building Construction Authority, Singapore
BERA	Building Environment Rule and Analysis, is another rule language developed as part of a PhD thesis [1].
BIM	Building Information Modeling, sometime it is used to refer to Building Information Model too.
BIMRL	BIM Rule Language, is the language designed to define a rule for an automated rule checking proposed in this research.

BIMQL BIM Query Language. It is a prototype query language built on top of BIMserver to perform create, read, update delete (CRUD) operations directly on IFC data in the BIMserver database.

BNF Backus-Naur Form, it is one of the main notation techniques in computer science for context-free grammar. This format is often used to describe the syntax of a language. It is also used in this research to define BIM Rule Language (BIMRL) syntax.

CG Conceptual Graph, a specific technique in the Knowledge Engineering domain for capturing knowledge

COBie Construction Operations Building Information Exchange

CORENET Construction and Real Estate Network, a major IT initiative to re-engineer the business processes of the construction industry to achieve a quantum leap in turnaround time, productivity and quality. It is an initiative led by the Ministry of National Development and is driven by the Building and Construction Authority in collaboration with other public and private organizations in Singapore

IBP Integrated Building Plan, an Architectural domain for the automated rule checking

IBS Integrated Building Services, an MEP domain for the automated rule checking

DRC Design Rule Check in the semiconductor mask design

EDM Express Data Manager, an express data management product from Jotne EPM, Norway

FM Facilities Management

GIS Geographic Information Systems

GSA General Services Administration in the USA, which manages all the Federal buildings in the USA

HVAC Heating, Ventilation and Air Conditioning

ICC International Code Council in the USA which owns and publishes the International Building Codes

IBC International Building Codes

IECC International Energy Conservation Codes

IDM Information Delivery Manual, is a methodology to capture and specify processes and information flow during the lifecycle of a facility. It is an ISO standard (ISO 29481-1:2010)

IFC Industry Foundation Classes, is a standard specification for the sharing of information between project team members and across the software applications that they commonly used for design, construction, procurement, maintenance and operations. IFC is an ISO standard (ISO 16739:2013)

LINQ Language Integrated Query, is a Microsoft .NET Framework component that adds native data querying capabilities to .NET languages

LOD Level of Development, is a specification that enables practitioners in the AEC Industry to specify and articulate with a high level of clarity the content and reliability of Building Information Models (BIMs) at various stages in the design and construction process

MEP	Mechanical, Electrical and Plumbing
MRC	Manufacturing Rule Check in the semiconductor mask manufacturing
MVD	Model View Definition, defines a subset of the IFC schema, that is needed to satisfy one or many Exchange Requirements of the AEC industry
OBB	Oriented (or Optimized) Bounding Box
OGC	Open GIS Consortium, an international voluntary consensus standards organization in the domain of geospatial or GIS
OSHA	Occupational Safety and Health Administration, is an agency of the United States Department of Labor
RDBMS	Relational Database Management System, is a database management system that is based on the relational model as invented by E. F. Codd, of IBM's San Jose Research Laboratory
SQL	Structured Query Language, is a special-purpose programming language designed for managing data held in a relational database management system
UML	Unified Modeling Language, is a general purpose modeling language in the field of software engineering
W3C	World Wide Web Consortium, is the main international standards organization for the World Wide Web
X3D	X3D is a royalty-free ISO standard XML-based file format representing 3D computer graphics. It is an open standard defined by web3D consortium (http://www.web3d.org)

XML	eXtensible Markup Language, is a markup language that defines a set of rules for encoding documents in a format which is both human-readable and machine-readable. It is defined by the W3C

SUMMARY

Rule checking for building designs has been universally acknowledged to have the tremendous potential to improve the quality and safety of buildings. Traditionally, it is viewed as a tool to assist the building code checking process. With the advances in BIM and its broad adoption in the past years, it is assumed that we are closer to the realization of an automated rule checking system for building designs with BIM as the enabler. BIM has brought us the integration of building information and its 3D intelligent objects into building models. It has bridged the gap between the theory, proposed in the 1970s, and the practice. With it, it is viewed that the task to develop an automated rule checking system becomes easier. However, despite many years of research and development in this area, we have not seen the progress we hoped to see. There is only one commercial application available today that is considered successful, i.e. Solibri model checker (SMC), and one turnkey implementation at the national level with CORENET ePlanCheck in Singapore. Even then, they have not really reached their full potential. Various research efforts have tried to address this issue. They focus on different aspects of the issues of rule checking. Some progress has been made, but the challenges stubbornly remain. In fact, the challenges actually increase with the rising popularity of BIM. No longer is it just about code checking compliance, but it has also widened to include all aspects of the building design, including the quality of building modelling, which is the focus of SMC, using BIM to improve the overall process of building lifecycles, specific domain best practices, etc. With such a wide range of requirements, an automated rule checking system appears to be still beyond reach.

This research has ambitious goals, and focuses on looking for a holistic solution to the challenges of an automated rule checking system for building designs. This is a response from real world experience in implementing the CORENET ePlanCheck system and the subsequent development to improve its implementation, along with a fresh approach to the same issue with the rule checking initiative at Autodesk using Navisworks. The experience indicates that it is insufficient to address the issues with piecemeal approaches, solving one part of the issue while leaving the others open. The reason is that rule checking requires an integrated approach. An efficient software tool can only work efficiently in tandem with efficient access to the data. Likewise, the BIM data must be accessible for the checking to be performed efficiently, but the schema design needs to match the expectation for rule checking. Additionally, there is great complexity in dealing with 3D geometry and spatial queries that are often inseparable from building rules, and with the enormous variations of the rules. To approach this problem holistically, this research looks at the problem from the macro level and zooms into the micro level in a systematic and integrated manner. At the highest level, rules need to be classified into four different classes of complexity as viewed from their potential implementation. The classification considers major components needed to solve the problem and the high level identification of the algorithms needed. Through this exercise, there is a recognition that the complexity of the rule implementation relates directly to the complexity of the checking requirements and the concept of derived objects that involve higher level semantic concepts. For the higher level classifications, higher level semantics need to be supported typically by some kind of geometry and graph engines. Most rules including building codes can be addressed with support for up

to class-3 rules. The increment of complexity between the classes is not linear and the chasm between classes can be wide. The first step of this research involves analyzing how to narrow the chasm and reduce the complexity, so that developing an automated rule checking system will be relatively easy with minimal development effort required, with the exception of very complex rules and those with very specific requirements.

The second step in this research is to analyze various rules and break down their structure into a systematic and computable form. The suitable form of documentation is in a form of knowledge engineering known as the Conceptual Graph (CG). Using CG, various rules are analyzed and broken down into their basic concepts and the relationship between concepts. The CG will force the breakdown of rules into their atomic rules, the most basic form that identifies what really needs to be checked. At the same time CG is also very useful in capturing knowledge of the rules that includes entities, properties, their relationship and functions that operate on the entities. This may even include entities at the higher semantic level. The CG is an important step in the rule analysis or rule interpretation since if done properly it will remove ambiguity often found in building codes. At the same time the outcome can be mapped directly to the data model needed and the design for computer implementation.

With the knowledge derived from the above two steps of analyses, three major components are identified to provide a backbone for an automated rule checking system that can simplify its implementation and use. All these three components represent the major innovations that this research advances, i.e. the use of a star-like schema to simplify IFC data for high performance queries, the approach to use multiple representations for geometry data, and a query-based extensible BIM rule language that

allows simple to complex rule definitions. There are other minor innovations that this research also produces, i.e. a non-overlapping octree based spatial indexing for efficient spatial operations, and the methodology and format to capture rule requirements using the Conceptual Graph.

The first major component is providing an efficient access to BIM data. While the IFC schema is good in facilitating data exchange, it is too complex for efficient queries of the data. The inherent tree-like hierarchical structure of the object based model in IFC and its select types make it hard to perform quick and dynamic queries. Therefore, this research explores an approach similar to the Data Warehousing concept that transforms building models from the IFC schema into a relatively flat, star-like database schema. This one way extract, transform and load (ETL) approach provides an optimized way to query the model. It essentially transforms the BIM data into a discoverable and query-able database.

As concluded from the analysis of rule complexity, alphanumeric based data in the database is not sufficient to address the most useful rule checking requirements. This leads us to the second component that involves an integrated support for the 3D geometry and spatial operators to the geometry. A similar concept that has been successfully done in the mostly 2D GIS domain is adopted and extended for 3D geometry. In this approach both the alphanumeric and the geometry data now reside in the same database and can be queried together using a standard SQL. This brings a positive implication that the BIM data is now open and accessible provided that the geometry data can also be queried and standard spatial operations can be applied. Efficient queries can be achieved using a 3D

spatial indexing based on Octree structure and multiple representations by denormalizing the geometry data into its basic and most often accessed data for rule checking.

Finally, the third major component in the rule checking system is the glue to all the other components and concepts, i.e. a domain specific rule language simply named BIM Rule Language (BIMRL). BIMRL makes use of the query-able BIM data in the database that is integrated with the geometry and its basic spatial operators. It takes a form that is identified from the study of the rule structure that follows the form of a triplet statement: CHECK – EVALUATE – ACTION. It is built on top of an SQL interface, but it is simplified for a specific purpose: to capture building rules as simply as possible. The goal is to make rule checking to be easy to define and run, even by rule experts who are usually not programmers. The language inherits the power and flexibility of the relational concept that provides simple to complex queries almost always at will. In this way the language will become an analysis tool to be used to perform any type of check on the building model. Recognizing that it is not possible to have a language that can perform all imaginable rules, BIMRL is designed to allow for an extension using an evaluation function. The extension will facilitate rule checking that involves operations that are harder to perform or for specialized checking requirements and add them as component plug-ins to BIMRL. They can take advantage of the query capability and rule structure in BIMRL without changing the structure of the language itself.

A software prototype has been developed as a proof that the concept is able to solve the problem of rule checking, which is what this research really aims to solve. The prototype involves the following capabilities:

1. The ability to query and check the building model without expensive and heavy technical or programming requirements.
2. The ability to perform checks on both the properties and geometry of the building model.
3. Flexibility that is built into the language to allow queries with open ended sets
4. The ability to perform class-1 to class-3 rules (at least part of class-3) without additional extensions).

Using the prototype software, this thesis demonstrates how effective the approach is when it is applied to real-world building models (which includes the typical architectural, structural and MEP elements) using several representative rules. This prototype covers all the aspects discussed in this research, i.e. the database star-like schema, its ETL, 3D geometry support and its spatial indexing, the associated multiple representations, graph support in the database, and the BIMRL language parser and simple user interface with support for visual reporting using an open standard X3D format.

This research demonstrates that the holistic approach to the automated building rule checking problem takes it closer to realizing the true potential of rule checking systems. The author believes that it can even serve as an analysis tool to provide a decision support system regarding BIM. With its accessibility by rule experts or even modelers, BIMRL opens up a lot more avenues for research and uses of rule checking and analysis on BIM data.

CHAPTER 1

INTRODUCTION

1.1 Purpose

This thesis aims to define a new method to transform BIM (Building Information Modelling) data into a simplified representation that can be stored in a database system and indexed for easy and efficient retrieval using a standard query language. The data concerned includes both non-graphic properties and the geometries. This thesis focuses on addressing the immediate application of this simplified representation namely for general use in the automated rule checking system for building designs. The rule checking system encompasses the range of simple user defined rules to more complex building codes.

1.2 Motivation

The motivation behind this thesis goes back all the way to the late 2000 with the development of an automated code checking system for Singaporean building codes, known as CORENET ePlanCheck system [2]. It was a complex system that was largely ahead of its time. It involved various developing standards and technologies such as EXPRESS and IFC (Industry Foundation Classes) from the IAI (International Alliance for Interoperability), which has now been renamed buildingSMART International. It also involved software-as-service technology that precedes the familiar concept of Cloud today, and proprietary technology stacks such as EDM (EXPRESS Data Manager) and its rule schema, 3D modeler using ACIS, etc. Without much precedence, the system was developed using many proprietary approaches that proved to be expensive since it involved trial and error, and also was tackled with a piecemeal approach that seemed to

make sense at that time. The project was officially completed in 2005, but it never really managed to be used widely since there were too many changes in the building codes specifications and in the IFC standard since the project began. There were also further developments and improvements of various software implementations that affected the entire ecosystem that CORENET e-PlanCheck depends on, such as the BIM authoring tools, and the evolution of the use of BIM and its practices in the industry over the years also led to its limited use.

Having the experience of how painful and expensive it is to develop such a system, the main motivation to work on this topic was to define a better way to allow a more generalized approach to access relevant data of the building model and to create a standardized language to capture the rule requirements and to execute it using a query based system to access the data. All these should be made as simple as possible that will minimize the need to develop a complex and proprietary software to achieve it. And better still if the rule experts are able to define the rules themselves without the heavy software development effort typically required to do so.

1.2.1 Challenges

Rule checking on a building model is a complex problem. Much research work has been done in this area for the past 50 years, but with limited success. More details on the topic will be discussed in CHAPTER 2. The complexity covers the building data, how it is presented and how easy it is to access the information, the rule specifications and requirements that present entirely different complexities with their construct and their limitless variations, and various semantic meanings that often are domain dependent. Most of the efforts in rule checking have focused on part of the complexity, which often results in a very limited capability to solve building rules. Some solutions are too complex that the cost of implementation becomes prohibitively high, or too rigid that

they are unable to adapt to even minor changes to the rules or minor variations of the rules.

1.2.2 Research Questions

Considering the complexity of rule checking and from the experience gained as the lead of the development of CORENET ePlanCheck, this research looks into the following research questions that guide the research, which lead to the conclusions drawn as presented at the end of this thesis. The research questions are:

1. What would be a suitable approach to deal with rule checking complexity and the large nature of variations of the rule requirements?
2. Is there a way that sufficiently transforms the relatively complex BIM data into a format that allows efficient query into the data?
3. What is a suitable format to capture the rule requirements that allows no or little additional translation steps into the computer implementation?
4. A standardized language (often called a Domain Specific Language or DSL) seems to be necessary to harness the data and to represent the rule requirements. Is there a universal structure and grammar that can represent rules that range from simple to complex rules such as building codes?
5. What would be the essential features that need to be represented in the language that will provide sufficient capability to meet a complex and large variation of the rules without the need to alter the grammar even with the future extensions?

1.2.3 The Research Scope

To address the above research questions in this research, the author takes a fresh and holistic approach by taking all the major components of building rules that contribute to the complexity as one single integrated issue. In this approach, rule checking is viewed from the perspective of its rule requirements, its structure, the demands on the data, the

availability of the data, and the means to access the data. The research limits the scope into a search for a generalized and simplified schema that represent the BIM data in a simpler form with the aim for efficient query. This is not just limited to its alphanumeric data, but includes also the more complex geometry data and its spatial related operators. This approach inevitably leads to a level of de-normalization of the highly structured BIM data, which in the scope of this research will be based on the widely accepted international standard IFC (Industry Foundation Class) maintained by BuildingSMART within the scope of IFC2x3 Coordination View 2.0 MVD. This is a very similar approach as in the field of Data Warehousing that seeks to achieve the same objectives to provide access to the data in the most efficient way that allows analysis and discoveries on the data, which otherwise is hard to achieve. This approach works well only within the scope of unidirectional data processing, i.e. the transformed data is intended for read-only purposes and not for updates. This fits well with rule checking applications since they need read-only access to the BIM data. In addition, since the aim of this research is to provide a generalized means for rule checking systems, and yet it is impossible to cover every imaginable rule existing, the research presents the theoretical support for the concept and it proceeds with a proof-of-concept using a few selected rules that are chosen to "prove" the effectiveness of the concept and how the concept is used in real examples. The examples are chosen to represent various sub-domains and those that significantly make use of the most components of the concept.

1.2.4 The Organization of the Thesis

This thesis is organized in a sequence that highlights a step-by-step progression of the development of the concept and its realization by means of the proof-of-concept. There are nine chapters that form the thesis and they are organized as followed:

- **Chapter 2 – Issues with BIM Rule Checking**

  The research begins with addressing the problem it tries to solve. In this chapter a brief survey of the previous related research works and projects are discussed. Then, an analysis of the efforts to-date is applied for the purpose of identifying the real need for rule checking research. With this, the systematic and holistic approach to the problem in rule checking is identified.

- **Chapter 3 – Classification of Rules Complexity**

  The first step in this research is classifying the complexity of the rules into broad categories that require different approaches to solve them. In this chapter, representative examples are discussed in detail to highlight the complexity involved and the components required to solve the problems are identified.

- **Chapter 4 – Analysis of the Logic Structure of the Building Rules Using the Conceptual Graph**

  Once the rules have been classified and the requirements to address the complexity are defined, detail analysis of the structure of the rules are applied to various rules from different genres to identify the general form of the rules. It uses a form of Knowledge Engineering approach to capture the rule requirements and breakdown their structure. The specific technique that is suitable for this purpose is the Conceptual Graph (CG) that will be demonstrated by applying it to several examples of rules.

- **Chapter 5 – Query-able BIM and its Schema**

  Having identified the complexity of rules and their structures, to successfully define an automated rule checking system, both building data and the rules need to be simplified and made accessible. Three major components to address that will be explained in the next three chapters. This chapter deals with

defining a suitable database schema that can be queried efficiently to support rule checking. The simplified schema takes a form of the Data Warehousing approach in the database domain that involves a transformation of the original data into a denormalized form that provides an identical view of the original data but with a very different structure. The main aim is to provide an efficient access to the data, which makes BIM data to be query-able or discoverable.

- **Chapter 6 – Multiple Representations for BIM Geometry Data**

  Having a simplified access to the objects and their properties is only part of the solution. The next is to simplify access to the geometry and to enable spatial operations. In this chapter, the concept of de-normalization of the 3D geometry into multiple representations in a database for the purpose of efficient query for use with rule checking systems is discussed. The approach taken in this research is described in detail with the strategy of indexing the geometry data in the same approach of de-normalization as the rest of the BIM data. The multiple representations stores the 3D geometry in the forms of triangulated mesh polyhedra, Octree cell decomposition, boundary representation with its indexed faces, and their simple bounding boxes.

- **Chapter 7 - BIM Rule Language (BIMRL)**

  To efficiently harness the de-normalized data structure, and to provide a simple query system, a domain specific language is presented. The language is named simply as BIM Rule Language (BIMRL). The main aim for the language is to provide a "shortcut" to the BIM data and allows rules to be defined in a relatively simple manner without much need for complex control structures typically applicable to a programming language. This language is a query-based language that makes use of a standard SQL (Structured Query Language).

- **Chapter 8 – Proof-of-Concept**

  In Chapter 8, a detailed description of the proof-of-concept is presented. The chapter begins with the detailed description of the prototype implementation and is then followed by the selected test cases. Four test cases are used to demonstrate how the BIM Rule Language can compactly describe the rules and how the query-able BIM schema supports such queries. The selected rules use all the major features described in this research including geometry, spatial operations, multiple sets, the use of multiple representations, and dynamic queries using graphs. They also represent complex rules with many known characteristics of checking types and operations typically found in complex rules. Two of the cases are taken from building codes representing architectural and MEP rules, one case from a best practice from hospital design, and one case involving dynamic queries on a circulation graph.

- **Chapter 9 - Validation**

  The results of the validation of the proof-of-concept is presented in this chapter along with discussions specific to each of the rules and their potential, limitations and effectiveness in solving the problem.

- **Chapter 10 – Conclusions**

  In this final chapter, discussions on what research questions have been addressed and the promises of the approach are presented. Possible further research of this approach is also discussed and recommended.

1.3 Additional Notes

There are various names that are often used to refer to rule checking, such as code checking, code compliance checking, rule checking. In this thesis, one uniform term that is generic will be used, i.e. rule checking. It encompasses code checking since code

checking is just another form of rule checking, albeit a complex one, that has links and connotations of laws and regulations.

CHAPTER 2

ISSUES WITH BIM RULE CHECKING

2.1 Why is Rule Checking Hard?

Buildings are erected for human beings to populate them and to live in them. Because of that, safety and comfort issues of buildings occupy an important role in designing and constructing a building. Building codes therefore have been with us as long as we have had a form of written rules. The earliest known form of the written building code comes from the codes of Hammurabi, the king of Babylon, around 2250 BC. It specified laws related to the building and safety aspect of buildings (clauses 228 – 233) [3]. Also, the book of Deuteronomy in the Bible prescribes a requirement for a battlement or a parapet to be built around the roof to prevent people from falling (Deuteronomy 22:8). The current forms of building codes are a culmination of many years of experience, trials and errors, and series of accidents or disasters, which cause building codes to fill up large volumes. They are also highly complex in most cases. With the increasing complexity, it is widely acknowledged that automating the process of checking would greatly improve the process and the assurance of the building designs. It is not surprising that even in the earliest days of computers, the idea of automating the code checking process was explored by Fenves in the 1960s [4]. The subject has been constantly and actively researched throughout the past 50+ years. Dimyadi summarized the efforts up to 2013 as shown in Figure 1 [5].

With the wider adoption of Building Information Modelling (BIM), there is significant progress in this area, but it is also faced with increasing challenges as new requirements emerge beyond the traditional code compliance checking. There is an increased demand for many faceted domain specific and best practices rules to improve

not only the quality of designs but also to improve productivity in the design process and further on for the construction process as well.

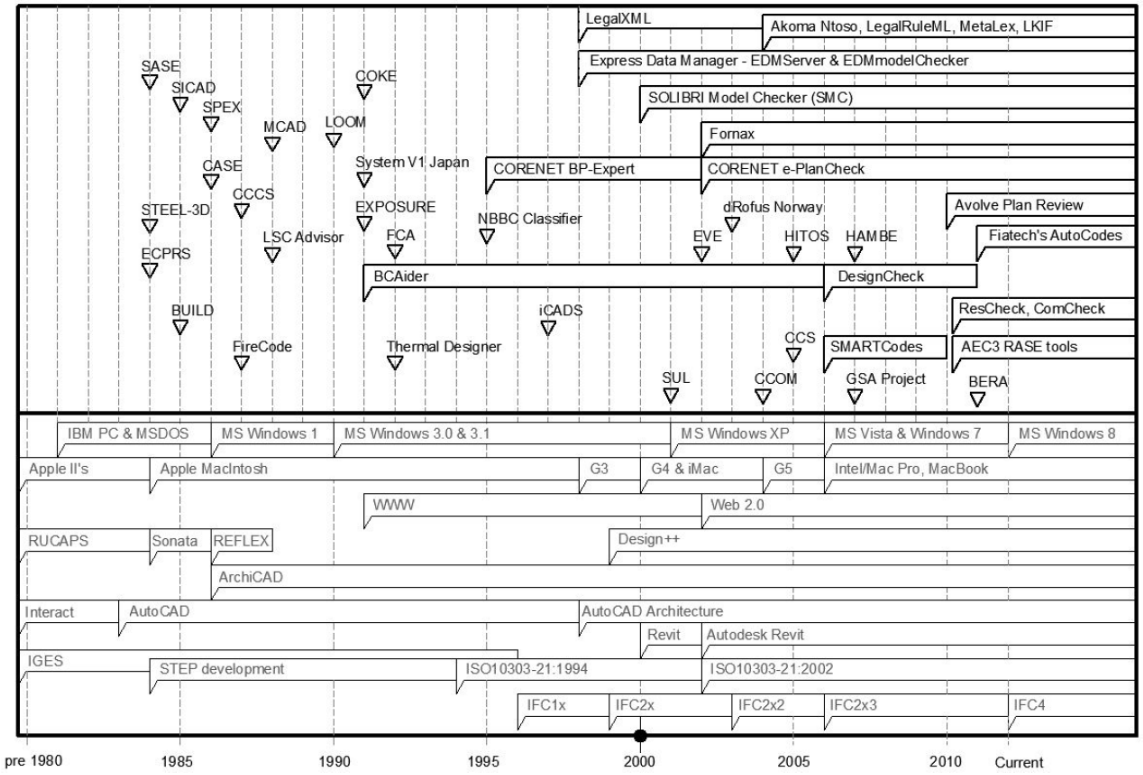


Figure 1 - Timeline of International Research into Code Compliance Checking [5]

2.1.1 Survey of the previous research on Rule Checking

As mentioned earlier, rule checking research has been ongoing for the past 50+ years. Eastman [6] and Nawari [7] provided a good coverage of the brief history of automated building code checking efforts. In this thesis, the focus is placed on what precisely the efforts focus on and what problem areas they try or managed to solve. From the literature survey and review of tools developed to address the problem that includes a handful of commercially available software on this subject, there appears to be several focus areas:

- The rule structure and its translation into computable forms
- Implementation approach of the computable forms
- Domain specific rules

- BIM data and the query system

Each of the focus areas will be expanded in more detail below:

1. The Rule structure and its translation into computable forms

With seemingly complex rule specifications and an assumption that BIM data contains all that is needed for rule checking, many efforts focused on analyzing the rule structure into more computable forms. Generally the rules are written in a human friendly form, especially in the case of building codes. This is rightfully an important step toward managing and reformatting the rules into a friendlier computable form of rules. Fenves introduced a decision table method in the 1960s [4, 8]. Hjelseth et al. introduced a RASE (Requirement, Applicability, Selection, Exception) methodology to tag the rules separating keywords into one of the four RASE categories. This work identifies and recognizes the general form found in building codes as demonstrated in the use of this methodology in the SmartCode project [9, 10]. Using such a tagging mechanism, building codes can be categorized based on the four categories without too much of the interpretative efforts, and it enables text search on the codes with the applications for automating relevant code searches for a manual compliance checklist. While Hjelseth proposed the use of this methodology into a computable form of rule checking by proposing the use of the IFC constraint model [9], it has not made much progress due to its dependence on the explicitly available data in the constraint model that is not supported by any application today. It also has not addressed the issue of the data and semantic representation of the data that maps the keywords into the BIM data. For example, part of the rule cited in the paper, ICC IECC 2006 502.5 Moisture control, states:

> *… The vapor retarder shall be* $\{blue\backslash\}$ *installed on the warm-in-winter side* $\{\backslash blue\}$ *of the insulation. …*

The {*blue*} tag represents one of the R (Requirements) of the rule to be checked, but it does not exactly deal with the semantically specific concept of "installed on the warm-in-winter side", which typically does not exist as explicit data inside BIM data.

In another track, El Gohary et al. took an approach using Natural Language Processing (NLP) to interpret and transform the rules into a computable form [11-13]. Like the RASE methodology, it does not account for the semantic meanings of terms and the automatic natural language processing will not adequately address the interpretative meaning of certain kinds of logic implicitly defined in the sentences. The approach alone does not solve the issue of rule checking complexity. Another approach is using sematic web and ontology to approach rule checking [14], which provides a promising way to define better semantics of the rules but still suffers from the same issue as the other approaches in the same category.

2. Implementation approach of the computable rules

Beside research work in this category, there are also commercial software developments that have achieved certain success in the automated rule checking implementations, i.e. Solibri Model Checker (SMC) and CORENET ePlanCheck that uses FORNAX as the engine. SMC is probably the only off-the-shelf and commercially successful implementation of BIM based rule checking that is widely used in the AEC industry. SMC addresses various rules that includes modeling quality checks with the emphasis on geometry, information takeoff that represents specific BIM data queries, and the implementation of at least partial building rules such as escape routes and disabled access rules [15]. FORNAX on the other hand focuses on complex building codes, starting with Singapore building codes that encompasses the subdomains Architectural and Building

Services (MEP) [16]. It has achieved limited success in Singapore and has been used in various pilot projects in the UK, Norway, New York City and recently in the Kingdom of Saudi Arabia [17, 18]. While SMC provides powerful checking functionalities, it remains a closed system that does not allow customization or implementation of new rules. FORNAX provides an API that allows developers to develop new rules, but it still requires considerable effort and domain knowledge to write one. Recent research in the UK used a similar approach to FORNAX to develop rule checking system for the Scottish building codes [19].

On the other hand, various research work into automated rule checking have been trying out various approaches, for example an Australian DesignCheck project completed in 2004 for the prototypes of automated code checking for Australian building codes using IFC and EDM Modelserver [20, 21], Beach et al. extended the RASE methodology and used DROOLS open source rule engine to implement the rules [22], Zong and Ding used the semantic Web OWL and SWRL technique and used JESS rule engine for its implementation. Nawari proposed the use of LINQ programmatic query language to implement rules [23] and Park used a rule base system CODE-MAVEN [24]. Choi took a similar approach as SmartCode tagging but used XML to compare the tags and the model exported from STEP P21 file that contains the BIM data [25]. None of the research is still active today and they achieved limited success due to the sheer number of possible paths and requirements for rules and their inability to extend the rules. The biggest factor was the lack of support for geometry and spatial operators to different rules. A very recent work by Preidel looks into an interesting idea of using a visual language similar to Grasshopper or Dynamo. It offers a potential use for defining rules without directly writing a computer program [26].

3. Domain specific rules

Similar to the above, several research articles that apply automated rule checking into a very specific problem domain have been published, for example its application in OSHA safety during construction rules [27], automated code-checking application for water distribution systems [28], the HVAC system [29], building envelop design [30], and integration with a thermal simulation system [31]. The Georgia Institute Technology under the guidance of Prof. Charles Eastman has successfully implemented GSA courthouse design rules for spatial programme and circulation rules [32], and also circulation requirements that led to the development of the BERA language [1]. While these systems work well within the domain they try to solve, they are restricted only to specific rules and generally are not extensible. BERA language offers the possibility of extensible language but it has not been developed further outside of the circulation rules.

4. BIM data and the query system

While much effort has gone into the rules and their implementation, the issues of the availability of the relevant building data and its accessibility have not been addressed adequately. EDM Modelserver is one such implementation commercially available and has been implemented in a pilot project in Norway, but it is limited to IFC data structure and lacks the support for geometry and spatial operations critically needed in many of the rules [33]. There are significant contributions in this area from Borrmann et al. who developed a spatial query system on the BIM data using an SQL extension, first with an Octree approach [34-39] and recently with a combination of an R-tree and Brep approach for more exact geometry [40]. Although this approach has the potential to open the possibility of access to BIM data with spatial operations that are often part of rule

requirements, it has not been fully expanded to deal with a real life sized building model and does not account for more complicated relational and spatial operators that are often required in more complex rules. It also does not account for the need to integrate the entire query system with the rest of the BIM data set efficiently.

The development of BIMserver and its open source model has offered an interesting opportunity for improved access to the IFC based BIM data in a way similar to the familiar database systems [41]. It does not contribute directly to the development of rule checking systems, but it provides supporting infrastructure that may enable the use of database queries to support the rule development by making it easier to access the data. The additional initiative to develop a prototype work BIMQL also adds into tools toward facilitating easier access to write an automated rule checking system [42].

2.1.2 Practical Insight from the Implementation of CORENET ePlanCheck

The author had the privilege to play a key part in leading the development of a comprehensive automated code checking system that started in late 2000 and was officially completed in 2005. A paper presented in a BuildingSMART meeting in Singapore highlights lessons learnt from the implementation effort [43]. Several very important lessons learnt that continuously shaped the thinking that led to this research are listed below:

1. The rule checking problem is a complex problem that must be addressed holistically in a way that covers not only the rules, but also the BIM data, the paths to access the data, geometrically and spatially enabled BIM data, an efficient query system, a generic structure for the development of rules, a practical way to document rule requirements in a compact and understandable way by

every stakeholder, and an ability to construct transient geometry that is often required as part of the checking logic.

2. In order for any effort to develop a rule checking system to be successful, the fact that rules have varying complexity and difficulty must be recognized and categorized accordingly to identify the real need to address the problems and to set the right expectation on what such a system is able to solve.
3. BIM data must be reliable and there must be a standard modeling practice and preferably a standard set of object libraries. For the standard to be effective, it must be well defined and must be able to be validated.
4. The analysis of the rules cannot be done in isolation, whether it be the isolation of one rule against another, or the isolation of rule structure outside of the requirements and semantic context of the rules.
5. While IFC is the standard that is widely used in the industry, there has to be a more efficient way to enable efficient access to the data without the burden of traversing a highly structured IFC schema.

2.1.2.1 FORNAX as a Rule Checking Platform

FORNAX, the platform used in developing CORENET ePlanCheck has addressed some of the issues above. It has implemented a layered architecture to provide a set of APIs that enables customization and development of new rules possible. The APIs are built using a FORNAX object model that allows encapsulation of data and methods into an object, which allows a neat access to many of the already implemented features and geometry functions (Figure 2). Figure 3 shows an example of what a FORNAX object looks like for a derived entity representing a staircase shaft. The staircase shaft is not just a collection of spaces in the vertical stack, it is defined as an object in an Object Oriented fashion that has both attributes and method just like another object such as a Space. For from a Shaft object, one can directly obtain all its boundary objects. From here a fire

rating can also be obtained, typically the lowest value of all the fire ratings of its boundary elements. Right from the start FORNAX has integrated the geometry engine and its corresponding spatial operators effectively extending the standard building element based concept in IFC to a much more semantically rich building object model [6, 44]. In this way, the system is able to provide some sort of self-correcting capability that shields the user from dealing with variations of input data, and "patches" data as much as possible underneath the APIs without user manual intervention as described by Solihin [45].

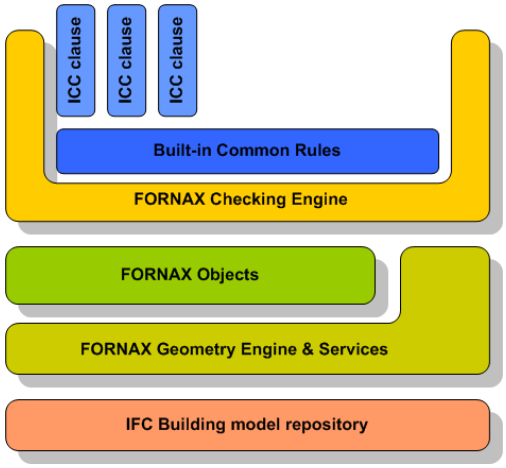


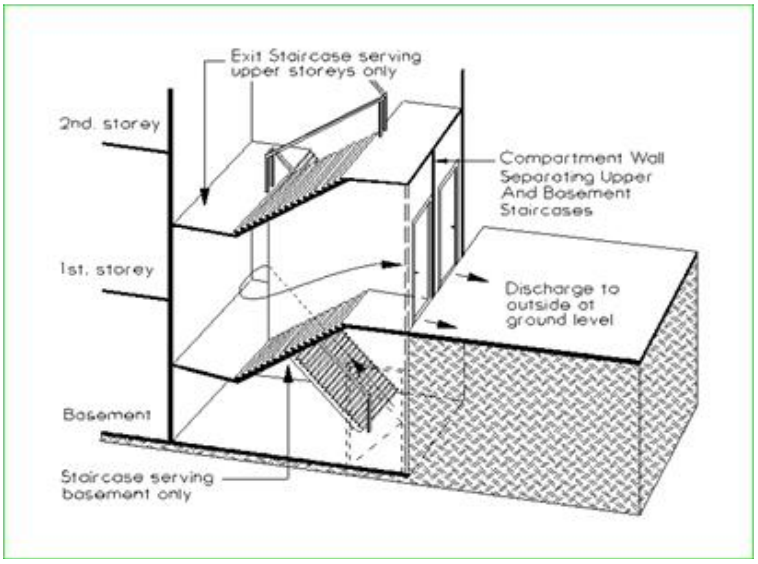
Figure 2 - FORNAX System Architecture

FORNAX therefore provides good access to the BIM data and the services to operate on the data, which includes geometry and spatial operators. The effort to develop automated rule checking is greatly reduced when using it as a platform. For example, 281 rules of the Integrated Building Services (IBS) that are MEP domain rules were re-written with FORNAX as the platform roughly in the period of 6 months, compared to a period of more than 2 years for Integrated Building Plan's (IBP, or Architectural domain) 167 rules, which started without a well-designed platform.

Even with carefully designed architecture, FORNAX is not yet a perfect solution to the automated rule checking. The shortcomings of FORNAX are:

1. Its dependence on assumed good quality of data. FORNAX depends on the data quality of the building model. A well-defined MVD may help this, but FORNAX was developed long before the concept of Model View Definition (MVD) was introduced in 2006 [46].
2. Development of a new rule, even if it is a modification to the already developed rule, still requires non-trivial programming efforts and often with the knowledge of geometry and solid modeling.
3. A model is still locked behind FORNAX APIs and therefore queries on the building model is still not a very simple task.
4. While FORNAX APIs offer a powerful set of features for rule development, the granularity of separation between data and checking logic is often still too coarse, making it necessary to apply relatively complex programming efforts. It has not been able to provide an easier access to the rule directly by the rule experts who are mostly not developers.

FORNAX is therefore suitable for dealing with complex rules, i.e. building codes, which generally require a large investment of time and effort. The use of it as a platform reduces time and effort significantly compared to developing it from scratch. But it also prevents the use of it for simpler rules. It is in contrast to Solibri Model Checker that deals with relatively simpler and well-defined sets of rules mainly for quality checking.



	Methods in FXExitStaircaseShaft	*Description*
Methods with low-level semantics (*available indirectly from IFC data*)	**IsPressurized**	Check if this exit staircase shaft requires pressurization
	GetFireRating	Get the fire-rating of this exit staircase shaft.
Methods with high-level semantics (*needs advanced geometry operations*)	**GetDischargeLevels** **GetNearestDischargeLevel**	Fetch all discharge levels from this shaft. Calculate the nearest discharge level through this shaft for a given storey.
	MinStaircaseWidth	Calculate the minimum width of the staircase in this shaft between any two storeys which this shaft serves.
	SwingDirection	Calculate the swing direction relative to this shaft for a given door.

Figure 3 - An Example of a FORNAX Object for the Exit Staircase Shaft

2.1.2.2 CORENET ePlanCheck Development Process

One of the critical process in any development of an automated rule checking system, especially for building codes, is the interpretation step [6]. This step took approximately 30% of the development time of each rule [47]. It resulted in detailed analysis of the rules that are captured in extensive documentation. Much of the documentation capture more detailed descriptions or explanation of the rules, the checking requirements, the objects in focus for checking, explanations of implicit assumptions and dependencies, checking logic and rule exceptions. Figure 4 shows a sample of such documentation produced by CORENET ePlanCheck project. The documentation contains several sections:

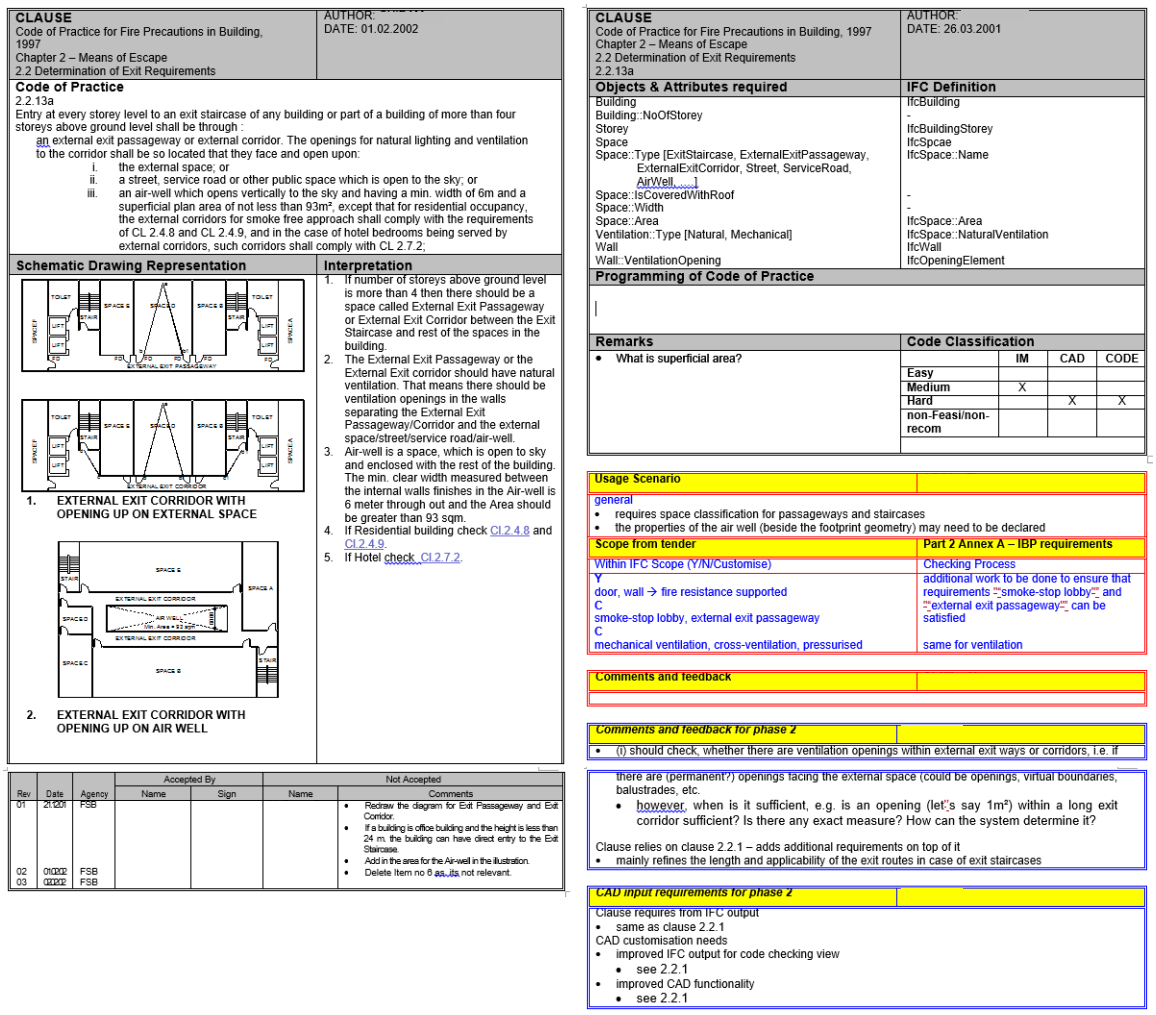
a. Rule header section, which describes the rule number from the original source and the author of the document plus date when the document was created.
b. Interpretation section. In the interpretation section, diagrams or floor plans are often used on the left column to aid understanding of the clause and to give the context in which the rule applies. On the right hand side, the text describing the interpretation is usually listed in a numbered list. The description may contain clarification of terms, checking conditions, checking logic, constraints and any other knowledge obtained from interviews with the owner of the rule.
c. Interpretation tracking section and approval or acceptance of the interpretation document.
d. An MVD-like section. It is called an MVD-like section since it was done prior to the definition of MVD. It captures what is required to be in the IFC model, this is in essence what MVD is. It also contains an early assessment of the level of difficulty.

The sections below (highlighted in yellow in the document) are internally used within the implementation team:

e. Usage scenario section. It is being filled by the domain expert to assist the development team in understanding the requirement, scope and scenarios for testing.
f. Comments and feedback section.
g. CAD input requirements section, which is intended to be used for the implementer's agreement when this MVD is implemented by the BIM authoring tool vendor.

The sample is arbitrarily chosen because of its completeness to represent the documents and its compactness in size. Many other documents are much longer than this.

As these are description-based documentation that have been done by various people, there was difficulty in maintaining a uniform quality of the documents and there were plenty of "lost in translation" cases. This requires a certain degree of redundancy to "interpret" the documents at every major step of development whenever handover from one team to the other occurred, for example from the rule analyst to the developer and from the developer to the tester.



CLAUSE
Code of Practice for Fire Precautions in Building, 1997
Chapter 2 – Means of Escape
2.2 Determination of Exit Requirements
2.2.13a

AUTHOR:
DATE: 01.02.2002

Code of Practice
2.2.13a
Entry at every storey level to an exit staircase of any building or part of a building of more than four storeys above ground level shall be through :
an external exit passageway or external corridor. The openings for natural lighting and ventilation to the corridor shall be so located that they face and open upon:
i. the external space; or
ii. a street, service road or other public space which is open to the sky; or
iii. an air-well which opens vertically to the sky and having a min. width of 6m and a superficial plan area of not less than 93m², except that for residential occupancy, the external corridors for smoke free approach shall comply with the requirements of CL 2.4.8 and CL 2.4.9, and in the case of hotel bedrooms being served by external corridors, such corridors shall comply with CL 2.7.2;

Schematic Drawing Representation

1. EXTERNAL EXIT CORRIDOR WITH OPENING UP ON EXTERNAL SPACE

2. EXTERNAL EXIT CORRIDOR WITH OPENING UP ON AIR WELL

Interpretation
1. If number of storeys above ground level is more than 4 then there should be a space called External Exit Passageway or External Exit Corridor between the Exit Staircase and rest of the spaces in the building.
2. The External Exit Passageway or the External Exit corridor should have natural ventilation. That means there should be ventilation openings in the walls separating the External Exit Passageway/Corridor and the external space/street/service road/air-well.
3. Air-well is a space, which is open to sky and enclosed with the rest of the building. The min. clear width measured between the internal walls finishes in the Air-well is 6 meter through out and the Area should be greater than 93 sqm.
4. If Residential building check Cl.2.4.8 and Cl.2.4.9.
5. If Hotel check Cl.2.7.2.

Rev	Date	Agency	Accepted By: Name	Accepted By: Sign	Not Accepted: Name	Not Accepted: Comments
01	21/201	FSB				• Redraw the diagram for Exit Passageway and Exit Corridor. • If a building is office building and the height is less then 24 m. the building can have direct entry to the Exit Staircase. • Add in the area for the Air-well in the illustration. • Delete Item no 8 as its not relevant.
02	01/000	FSB				
03	00/00	FSB				

CLAUSE
Code of Practice for Fire Precautions in Building, 1997
Chapter 2 – Means of Escape
2.2 Determination of Exit Requirements
2.2.13a

AUTHOR:
DATE: 26.03.2001

Objects & Attributes required	IFC Definition
Building	IfcBuilding
Building::NoOfStorey	-
Storey	IfcBuildingStorey
Space	IfcSpcae
Space::Type [ExitStaircase, ExternalExitPassageway, ExternalExitCorridor, Street, ServiceRoad, AirWell, ...]	IfcSpace::Name
Space::IsCoveredWithRoof	-
Space::Width	-
Space::Area	IfcSpace::Area
Ventilation::Type [Natural, Mechanical]	IfcSpace::NaturalVentilation
Wall	IfcWall
Wall::VentilationOpening	IfcOpeningElement

Programming of Code of Practice

Remarks
- What is superficial area?

Code Classification

	IM	CAD	CODE
Easy			
Medium	X		
Hard		X	X
non-Feasi/non-recom			

Usage Scenario

general
- requires space classification for passageways and staircases
- the properties of the air well (beside the footprint geometry) may need to be declared

Scope from tender	Part 2 Annex A – IBP requirements
Within IFC Scope (Y/N/Customise)	Checking Process
Y door, wall → fire resistance supported	additional work to be done to ensure that requirements ""smoke-stop lobby"" and ""external exit passageway"" can be satisfied
C smoke-stop lobby, external exit passageway	
C mechanical ventilation, cross-ventilation, pressurised	same for ventilation

Comments and feedback

Comments and feedback for phase 2
- (i) should check; whether there are ventilation openings within external exit ways or corridors, i.e. if there are (permanent?) openings facing the external space (could be openings, virtual boundaries, balustrades, etc.
- however, when is it sufficient, e.g. is an opening (let's say 1m²) within a long exit corridor sufficient? Is there any exact measure? How can the system determine it?

Clause relies on clause 2.2.1 – adds additional requirements on top of it
- mainly refines the length and applicability of the exit routes in case of exit staircases

CAD input requirements for phase 2
Clause requires from IFC output
- same as clause 2.2.1

CAD customisation needs
- improved IFC output for code checking view
  - see 2.2.1
- improved CAD functionality
  - see 2.2.1

Figure 4 - A Sample of CORENET ePlanCheck Rule Interpretation Document

2.2 A Survey from the Manufacturing Domain

In many ways, the AEC domain is similar to the manufacturing domain and many of the processes and technologies have been adapted from the manufacturing domain as

well. It is logical then to ask the question: what can be found in the manufacturing domain in respect to automated rule checking? With this question to start with, a literature survey was conducted, with a rather surprising result: i.e. there is not much relevant literature that discussed rule checking use in the manufacturing industries, with the exception of the semiconductor sector. The main focus of the manufacturing industries when dealing with rules is related to Design for Manufacturability (DFM). It is also true for the semiconductor sector where the design of an integrated circuit (IC) is checked for quality and for issues that may affect its manufacturability of its masks. It does generally in two steps: during design (Design Rule Check or DRC) and during preparation for manufacturing (Manufacturing Rule Check or MRC) [48]. In both cases, the focus is mainly in terms of its geometry and in 2D (Figure 5).

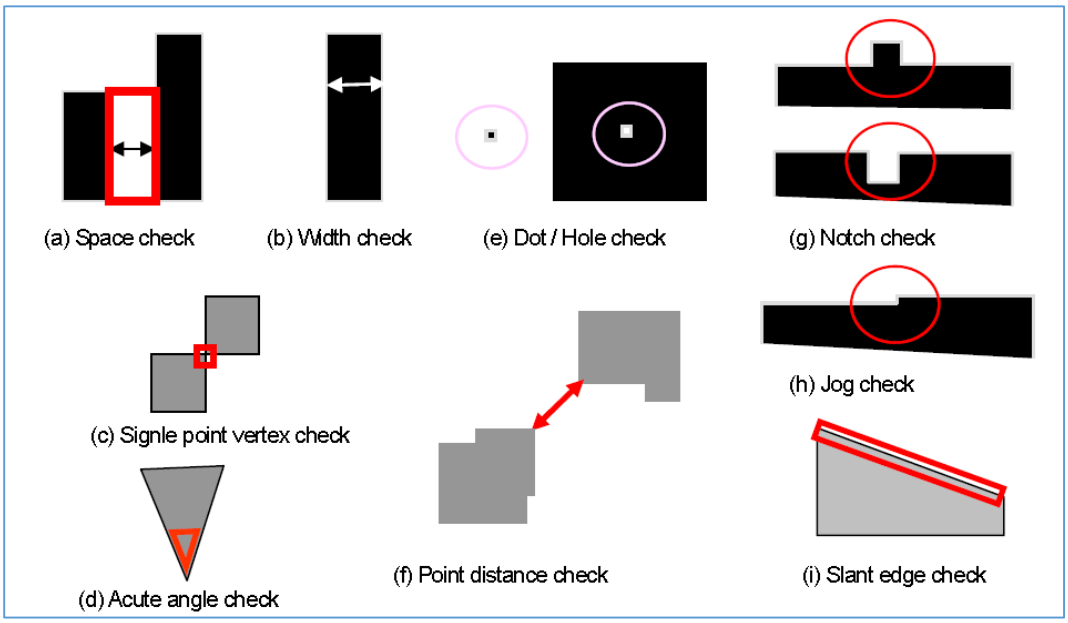


Figure 5 - Several Examples of Basic Rule Check from SmartMRC [48]

A further survey on the Siemens NX automated rule checking tool Check-Mate that works on 3D models reveals a similar characteristic, which deals with lower level semantic models focusing on geometry, standards, etc. Some classes of the built-in checks are highlighted below [49]:

- Model quality check (geometry based), for example:
    - Check a tiny geometry
    - Check self-intersection
    - Check geometry and topology of surfaces
    - Check tolerance
    - Check well-formedness of a geometry, e.g. sheet can be thickened, blend is correct
    - Check attribute values
    - Check segments, edges and curves
    - Etc.
- Assembly, file and drawing checking. For example:
    - Check completeness of assembly part drawings
    - Check all components that exist in an assembly
    - Check view setting in a drawing
    - Check consistency of manually entered dimension value
    - Various reporting functions, e.g. count number of features, geometry expressions in a part file
    - Etc.
- Library of standard checks, for example:
    - Checking layer standard
    - Etc.

NX boasts more than 300 pre-built rules and allows users to customize their own rules in addition to the built-in rules. Like other MCAD (Mechanical Computer Aided Design) applications, one category of checks in NX is geometric constraint checks that are important in design and drafting. Figure 6 shows an example of NX Check-Mate output.

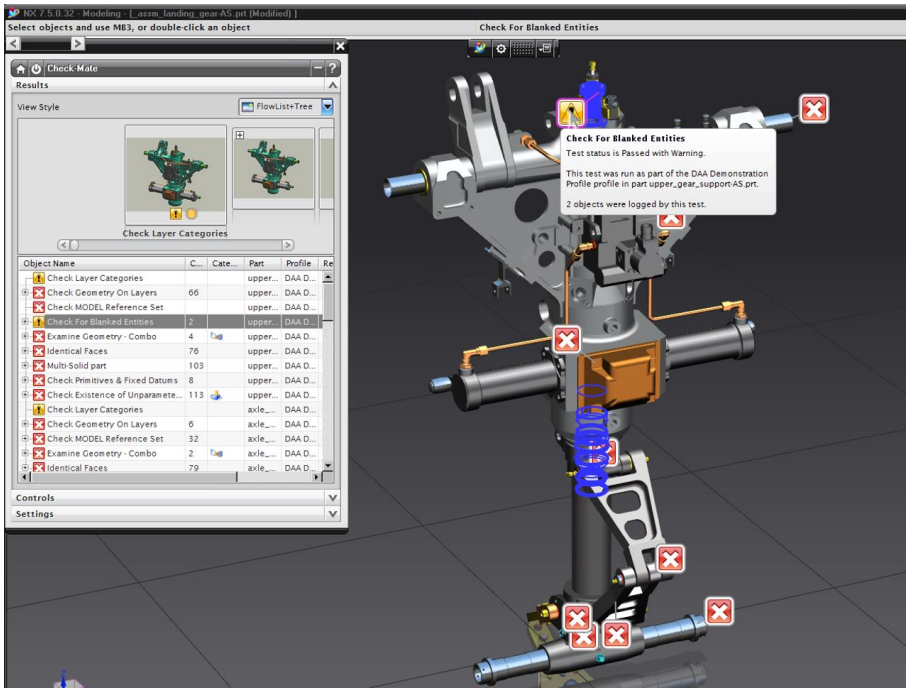


Figure 6 - An Example of Siemens NX Check-Mate Rule Checking

Rules to automate the checks for IFC model quality similar to the pre-built rules in NX for geometry and "standard" data have been proposed in [50]. It covers more than the current known IFC certification tests, but it is not in the scope of this research to cover this topic. This subject is actively being researched by Eastman's group in the Georgia Institute of Technology. Some of the issues have been covered by automated certification tests and by applications such as Solibri Model Checker.

The survey concluded that while BIM rule checking is a constraint check in a broad sense, it differs significantly in that the rules mostly operated on a much higher level of semantics or abstract domain specific concepts. For example a relatively straightforward rule from a hospital design best practice [51-53]:

> *"Every patient room shall be visible from a nurse station".*

It involves several abstract concepts such as patient room, nurse station, and the implicit concept of visibility where the source and destination positions need to be defined based on relevant objects tagged to the first two concepts. Figure 7 illustrates this rule.

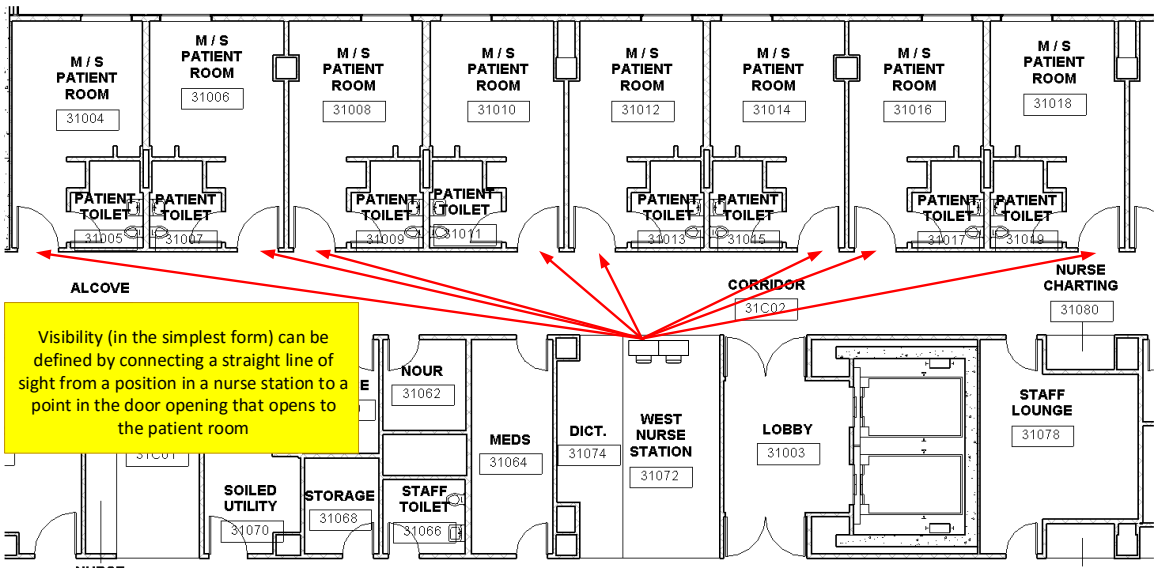


Figure 7 - An Example of BIM Rule Checking Requirement

This relatively straightforward rule requires several abstract concepts:

- A common circulation access for both the nurse station and the patient rooms
- A concept of access into the patient room in form of a door opening that connects the common circulation space and the patient room, or a view into the patient room by means of a glass window.
- A concept of line of sight that typically is not present in the model and needs to be derived from the primitive data extracted from the relevant objects, in this case a position within the nurse station and the center point in the door opening.

Many other rules especially those from building codes consist of heavy requirements for the abstract concepts specific to the AEC domain and its subdomains. In most cases the information is not available directly in the model. It is usually also not available to be derived meaningfully until the design reaches a certain level of completeness when all the related design objects are in place.

The survey concluded that apart from superficial similarities between automated rule checking in the Manufacturing domain and in the AEC domain, the real needs are in the domain specific abstraction semantics incorporated in the AEC rules.

2.3 Special Problem Requires Special Approach

This research is essentially a response to a combination of the broad base knowledge of what the BIM based rule checking requires, the failures of the current research efforts to develop a more generic form of a rule checking system, practical lessons learnt from the actual implementation in CORENET ePlanCheck, and the survey of automated rule checking from the manufacturing domain. The current research and effort have not sufficiently addressed the problem holistically and solve only part of the problem. This research aims to address the problem from all three major aspects of the rule checking issues:

1. Addressing the complexity of the rules by systematically analyzing the logic structure of the rules and identifying the higher level semantic or abstract requirements from the BIM data.
2. Addressing the need to have an efficient way to access and query the BIM data for a wide range of queries that often are difficult to predefine. This includes the often-neglected requirement to access the geometry and its relevant spatial operators.
3. Addressing the need to provide a relatively simple interface to access the data and define the rules. Ideally the interface should be easy enough for rule experts who are not necessarily computer programmers to define the rules and to examine the outcome.

CHAPTER 3

CLASSIFICATION OF RULES COMPLEXITY[1]

Rule checking covers a broad scope within the AEC Industry and its practices that covers the entire lifecycle of the building. It has expanded beyond the traditional requirements for building code compliance. Eastman [6] suggested that besides building regulatory code-checking, more specialized types of rule checking such as clients' requirements and requirements for specific building types are also emerging. In general, the scope of the rules fall into the following categories:

1. Checks for well-formedness of a building model. This group of rules concerns primarily syntactic aspects according to the set of standards or prior agreed set of required conditions for the IFC or for other model views.
2. Building regulatory code checking. This focuses on compliance to well-defined or usually prescriptive building codes or regulations.
3. Specific client requirements. Examples include requirements for hospital design or GSA courthouse design.
4. Constructability and other contractor requirements. These often involve temporary objects or conditions present only during the pre-construction process and during construction, such as formwork and shoring.
5. Safety and other rules with possible programmed corrective actions. These support decisions and help automate the search for potential dangers to workers during construction and also maintenance staff during operation.

---

[1] *The content of this chapter has been published as a paper in Automation in Construction vol. 53, May 2015 under the title "Classification of rules for automated BIM rule checking development"*

6. Warrantee approvals. The post-construction model is checked for issues that may affect the warrantee or cost to maintain. These requirements may need to be combined with actual site inspection. Roofing systems are an example. The tool however will provide assistance for the inspector to focus on potential issues of the design and construction.
7. BIM data completeness for handover to the facilities management (FM). Since the role of BIM changes during the lifecycle of the construction process, downstream requirements such as for FM are often not considered earlier in the process (to the detriment of the facilities manager). Rules to check such completeness at the end of different contract phases will be increasingly required such as requirements defined by COBie and the other families of Information Exchange (IE).

3.1 Re-Use of BIM Rule Checks

Looking at the above range of potential uses, it is apparent that rule checking has an extensive range of applicability. The methods used for specific building code checking may also be applicable to safety or constructability checking. Significant benefits accrue if rule checking can be organized generically according to the type of checking software infrastructure required, rather than by application area. This more abstract approach allows researchers to target critical needs across application areas and to identify issues benefitting the whole rule-checking area. The key principles that guide us in organizing the rule-checking scope are motivated by the following ideas:

1. Potential re-use of rule structures and clauses. Patterns exist in many rule structures and clauses even though they entail detail variations. The extraction of such patterns is challenging because it requires experience with different rule applications and inductive reasoning. This usually happens at the rule interpretation step [6], where rule experts analyze the rule and discover hidden

assumptions, dependencies, ambiguities and exceptions to define the exact rule requirements suitable for software development. Re-use of such patterns significantly improves coverage of the range of rules that can be implemented. This will in turn reduce cost of such implementation efforts and add to the logical structure that the rules can be based on.

2. Existence of best practices in specific areas especially in the areas where rules are not well-defined, but commonly accepted best practices exist in the industry. Many examples exist in geometric modelling, for example walls must touch slab below, externally exposed slab and internal slab should be modelled as separate slabs, all parts of the building should be "filled" with space objects for programmatic and floor area assessment, all spaces should be connected except for small spaces used for service ducts [15, 32, 54].
3. Strategies for closing the gap between the terms used in rules to be checked and the information explicitly represented in the target building model. The gap involves deriving new data that is represented implicitly in the provided model. It also involves the development of terminology for types of rules and their classification using a publicly available open standard such as IFC. IFC has been steadily accepted as a standard in the industry and is the only open and relatively mature standard supported today by major BIM applications. A standard way for representing building model data is crucial in developing any stable rule checking application.

3.2 Level of Development (LOD) and Model Views (MVD)

There are two major parts that a rule checking system must deal with. The first is the building model, and the second is the rule definitions. Building models are large datasets, even for medium-scale buildings. There is no rule or class of rules that applies to the entire set of building model data. Each rule or class of rules applies to a subset of

the data. Development of Model Views that represent the appropriate subset of data required for an exchange is critical in defining the exact rule checking requirements that the populated models satisfy. Model Views are a standard methodology proposed by Hitanen, and BuildingSMART has adopted it as a standard methodology in the implementation of IFC based projects [46, 55, 56]. Model views serve as "contract" documentation for both the modeler and the implementer. It is a kind of handshake protocol that allows expected meaningful requirements, implementation and results. Explicit definitions of model views relies on the inferred rules that are associated with the model view. It typically includes the types of geometry desired by the receivers, the critical variables for the use case, and the restrictions of entity subtypes of relevance in the use case. Such rules are sometimes referred to as implementer's agreements.

Another important factor affecting building models is the level of development of the building model, or its LOD. The LOD was developed by BIMForum as a specification to articulate with clarity the content and reliability of BIM at different levels of development [57]. Different objects and features have different LODs during the phases of a project. Each object's LOD monotonically progresses, but at varied rates. For example, the rebar detailing and layout progresses at a slower rate than the formwork because the formwork defines the constraints that determine the reinforcing layout. Conversely, the phases of the project lifecycle have different LODs, which identify the structure and detail that rules can apply meaningfully. The higher the LOD is, the more detailed information within the model is expected. There are currently six levels of LOD: LOD 100 mainly requires objects in graphical representation, LOD 200 adds approximate quantities, shape, location and orientation with possibly non-graphic information attached, LOD 300 requires more specific systems, objects or assembly in term of quantity, size, shape, location and orientation with possibly non-graphic information attached, LOD 350 adds requirements on interfaces with other building systems, LOD 400 contains more detailed information required for fabrication, assembly and

installation, and LOD 500 is a field verified representation. For building models at the design development phase, which is the most typical stage where a building model is complete enough for code compliance submission, LOD 300 or LOD 350 is generally sufficient. For this reason, it is expected that initially LOD 300 will be assumed, but over time rules will evolve to be able to work at least partially with a lower level of LOD. Fabrication issues will apply to higher level LODs. Requirements for rule checking may work harmoniously with systems at varied LOD. For our uses, LOD should be at the lowest level adequate for rule checking.

The second major part of rule checking is the rule definition. Today, the rules are typically written in human-oriented languages that require significant domain knowledge in order to "interpret" them into a machine interpretable manner. There are many ways of approaching the interpretation, as mentioned earlier, but most rule checking studies focus merely on the language representation of syntax and grammar of the rules. In practice, expert knowledge is often required to interpret the meaning or semantics of the rules: the intent, base and hidden assumptions, assumed general knowledge of the subjects, and dependencies with other rules. Experience in CORENET ePlanCheck, which used a logic-based interpretation method [6], shows that the interpretation is crucial in transforming the rules that are often ambiguous into more precise definitions, thereby removing the ambiguity and clarifying checking concepts or principles. This is usually achieved by asking a series of questions. During the process of interpretation, implicit assumptions or expectations are discovered that help to complete the understanding of what needs to be checked. Too often this fact is not rigorously considered due to the practical challenges in looking through all possible rules and in anticipating the complexity in terms of their implement-ability. This usually leads to a sampling of relatively low complexity rules and using them to prove the approach would work before extrapolating it to the rest. This approach may lead to gross underestimations of the complexity involved in the implementation of more complex rules, as suffered by the

CORENET implementation team [43]. To illustrate the issue, three examples below show the existence of implicit assumptions that will need expert knowledge in order to interpret them:

Table 1 - Example of ambiguity and implicit knowledge for requirements in building codes

Clause ref.	Description	Questions during interpretation process
Reg. 44(1) BCA	Protection of staircase and staircase landing Every staircase or staircase landing shall be protected on any side overlooking an air-well, courtyard, void or external open space by either a railing, parapet or balustrade capable of resisting the lateral loading as specified in the Table 4 of the fourth Schedule	1. What is the criteria when the protection starts to be required? What is the height of the protection? How about the shape of the protection (especially when there are gaps)? 2. What is exactly defined as "overlooking"? Is a full glass wall considered "overlooking" and therefore requires additional protection, or can the glass wall itself be considered a protection? 3. How far can a protection, e.g. railing, be from the edge before it is no longer considered a protection to that edge? 4. If the edge has a gap to the adjacent edge, how large a gap is allowed before protection is needed?
Code of Practice on Sewerage and Sanitary Works; Part 3.1 Sanitary Drainage Systems	3.1.3.6 Inspection Chamber and Ventilation 3.1.3.6 (b) i) The first inspection chamber shall be ventilated except when there is a discharge/ventilating stack of not less than 100mm diameter or where provision of the ventilation stack would cause odour nuisance to the surroundings.	1. How do we determine the first inspection chamber? 2. What qualifies as "ventilated"? 3. In the case of a stack, where can it be located relative to the chamber? 4. What defines when a ventilation stack would cause odour nuisance?
BCA Lighting requirements (F)	The aggregate light transmitting area for each room must not be less than 10% of the floor area of the room to be lighted	1. Unstated requirement: There is a need to consider the indirect light coming through windows from adjacent "buffer" space, such as veranda, balcony, wash areas, terrace, or corridor

The problem is how to address the domain expertise assumed in the interpretation of the rules – this can only be done manually today. Rule interpretation is a significant step in the process of rule checking. In CORENET, it took approximately 20 - 30% of the overall effort. This was not a small effort but this process allowed a one-time investment for good rule checking that led to automation of the manual effort. The same conclusion

was observed in phase 1 of the AutoCode project that Fiatech undertook: manual rule checking is largely inconsistent because of human judgement that fills in the ambiguities, incorporating experience and unwritten local adaptation of the rules [58]. The phase II of rule checking is to generate consistent, precise and quantifiable conditions and constraints for each rule [59]. A step in the right direction are the recent trends using Ontology and Semantic Web approaches to systematically capture domain knowledge embedded into rule definitions that transform the rules into a consistent representation suitable for the software engineers [11, 14, 60-64]. However, both traditional logic based interpretation and Ontology based interpretation do not remove the complexity in terms of actual checking requirements represented as predicates or functions that need to be codified.

In the actual codification of rules, one has to consider the trade-off between requiring most of the required data to be carried inside the model and entered by the user and using the computer program and logic to derive the new information. On one side, requiring the data to be in the model often dictates significant user input effort. This process will overburden the designer and it will be prone to errors and inconsistencies. On the other end deriving the information from basic BIM data inside the rule implementation will significantly increase the complexity in terms of the implementation [43]. For example, calculating the area and volume of a Residential Unit that includes a group of spaces and everything else in between. It involves many tasks that start with the spaces, finding all objects (mainly walls) bordering the spaces through the space boundary relationship, clipping the walls if they extend beyond the unit boundaries, and including the area of the holes in the spaces (for example, columns). Even with derived information, the responsibility of the designer to input minimum information may still be required, which should be consistent as the derivation rules in the Model Views.

3.3 Dealing with Arity Issues

Rule checking systems are hardly a simple one question one answer condition, even in its simplest form. Rule conditions can be satisfied with exactly one criterion, at least one of many possible criteria, or multiple/all criteria at once. Without consideration of the arity issue, a rule checking system will suffer from a large number of false negatives, which will become noise in reporting the exact checking results. This needs to be carefully avoided because even a single false negative for one rule usually will be multiplied to many; the object being checked inside the Building Model typically numbers in the hundreds if not thousands for reasonably sized buildings. The rule checking system must be smart enough to report only the real negatives according to the appropriate criteria defined for the rule.

3.4 Dealing with Combinatorial Issues

As rule checking progressively moves toward more complex requirements, it no longer simply deals with cases of returning True/False conditions on a straightforward parameter vs. value check, but rather evaluating combinatorial issues. It needs to deal with multiple possibilities where each possibility may lead to a different solution path. In a large building model, the combinatorial problems may easily grow into a large and hard-to-manage scope, often based, for example on space occupancy and type of structure. Clearly defining boundaries that make sense in narrowing down potentially large combinatorial conditions becomes critical, especially in the more advanced classes of rules that will be described later. Domain expertise and experience in dealing with rule checking will be extremely important in such cases. A simple example of combinatorial issues that rule checking needs to deal with is given in the appropriate example under the Class-2 rule classification for building codes from IBC 1008.1.8. (Section 3.7.1)

3.5 Rule Classification

In the end, all the data in IFC reduces down to attribute values. These may be material type, reference to a geometric placement, or x,y,z point locations. Considering this point, the fundamental operation is the reading of values in the model structure. The model structure can be simple or complex, such as when geometry is concerned. Based on the above discussion, this chapter classifies various rules according to the complexity of the rules processing. It also assumes in the examples the processing of an IFC model with the LOD 300, and using the "standard case" property sets and structure according to the basic MVD appropriate to the examples being cited. In many of the cases, they follow BuildingSMART's Coordination View 2.0 [65] and its extension, code-checking MVD that is used for CORENET ePlanCheck [66].

It is useful to distinguish four general classes of rules, described in more detail below. The description is organized by first explaining the definition, typical uses of the rules, followed by an explanation of the complexity and suitability of the tools or techniques required. When appropriate, one or two examples of rules are presented to illustrate how the rule checking is typically processed. The examples given in this paper represent one possible solution to address the specific rule discussed. This chapter does not go into the details of algorithmic structure of the rules. The examples rather are presented to highlight the complexity involved and to clarify the basis of classification of the rules.

In Table 3 a list of several other representative examples can be found, drawn from various sources commonly applied in the AEC industry. In Table 4 a summary of the list of applications both from research domains and commercial products that fit into solutions that match the classes is presented.

3.6 Class 1 – Rules that require a single or small number of explicit data.

This class of rules checks explicit attributes and entity references that exist inside the BIM dataset. Typical uses of this class of rules are:

a. the required attribute settings of entities for correctness checking
b. Some simple building code checking, for example “Egress doors shall be of the pivoted or side-hinged swinging type” (IBC 1008.1.2) [67], and “every building comprising 5 or more stories above the ground level shall be provided with one or more passenger lifts” (BCA Reg 53(1)) [68].
c. Some model view business rules in MVDs restricting options allowed, for example Name and LongName attributes, which are optional in the IFC schema for IfcBuilding, are mandatory in the Basic FM Handover MVD [69].
d. Attributes and attribute values that are essential for consistent derived values required in other classes of rules. For example, to be able to derive a concept of an apartment unit, all spaces within the unit must be members of an IfcZone and the IfcZone must be named and classified according to the standardized naming convention or classification.

3.6.1 Degree of complexity/required tool

At this level, information is explicitly available from the model either directly from the entities or its associated properties with other entities using the explicit relationship entities (Figure 8). Frequent tools used in supporting this level of rule checking are IFC toolkits that are available both commercially (for example EDM IFC toolkit from EPM Technology http://www.epmtech.jotne.com/, EuroSTEP toolkit, etc.) and various open source alternatives (a list is available from http://www.ifcwiki.org/index.php/Open_Source).

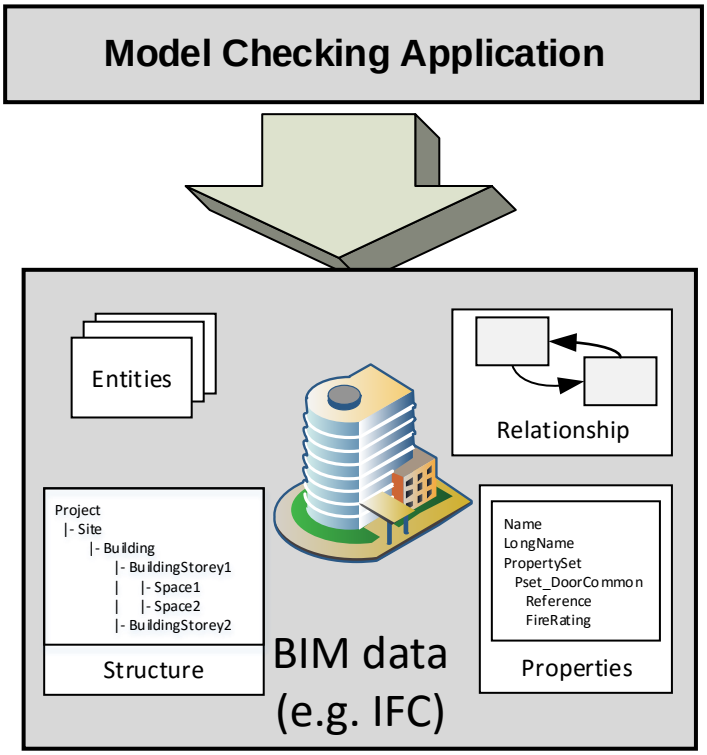


Figure 8 - Diagram for Typical Application Implementing Class-1 Rules

One example on how this type of rule can be processed:

"Fire Walls Must Have Correct Wall, Door, and Window Types" [SMC – Building Codes rules] [15].

To process this rule, the model is queried for well-defined entities (Wall and its components, Door, Window) and their relationship. The Fire Rating property to be checked is obtained using explicit relationships from each of the entities being processed. All entities and relationships are usually explicit in the model and the rule is able to follow the links in a straightforward manner.

3.7 Class 2 – Rules that require simple derived Attribute Values

Checks are based on a single value or a small set of derived values. This class of rule checks derive attributes or values but does not generate new data structures. This

class of rules involves a trade-off between requiring the user to derive the data vs. rule checking-derived data. The rule checking-derived data has a clear benefit to the industry, which needs the work to be done only once across all applications instead of including similar data derivations in each BIM platform.

3.7.1 Degree of complexity/required tool

At this level, implicit relationships are often required to fulfil the checking requirements. BIM models do not usually capture this explicitly and the program needs to derive the value from the basic BIM model data and relationship (Figure 9). Certain arithmetic or trigonometric calculations, such as finding the straight line distance between two points or determining if a point is inside an enclosed polygon, may be involved in rules of this class. Many IFC supporting tools are already capable of trigonometrically creating simple geometries that can be used to support such calculations without the need for a sophisticated 3D solid modeler.

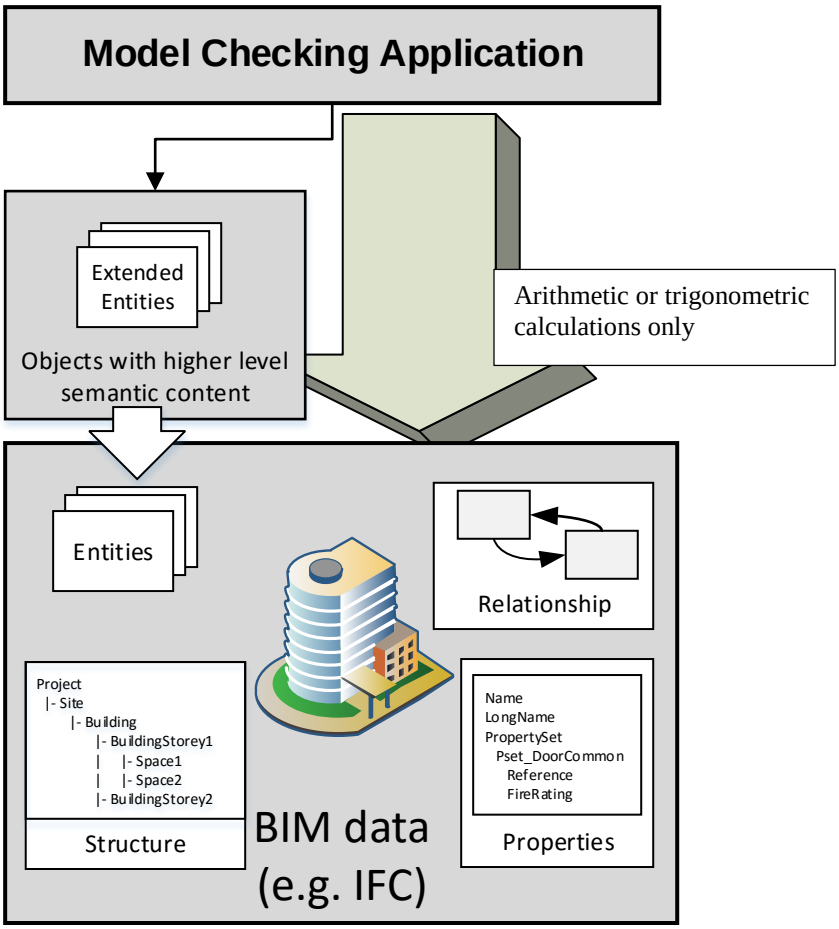


Figure 9 - Diagram for Typical Application Implementing Class-2 Rules

One example of this class of rules is ICC IBC 2009:

"1008.1.8 Door arrangement. Space between two doors in a series shall be 48 inches (1219 mm) minimum plus the width of a door swinging into the space. Doors in a series shall swing either in the same direction or away from the space between the doors."

This rule has two sub-rules that need to be tested independently: a rule to check whether the gap distance between the two doors is not less than 48" (1219 mm) after reducing it with the width of the door leaf opening into the space, and a rule to check whether the doors in series open in the same direction or if both open away from the space.

The first step in solving this rule requires an analysis of what is implicit in this code, which involves the definition of what constitutes doors in series. The illustrations given for the clause focuses only on the obvious (Figure 10) that assumes implicitly that a human expert will interpret the data by visual inspection to identify the cases of doors in series. Computer programs on the other hand require a precise definition to evaluate and distinguish the model from any number of possible configurations. Figure 11 shows several possibilities of different configurations of doors in series. This is not an exhaustive list of configurations, but it highlights the need to have a generic algorithm that can assess all the possibilities and constrain them into a reasonable definition of what makes doors in series. To illustrate the example one particular configuration is selected and the generic algorithm that can be used to handle this issue is described as illustrated in the diagram in Figure 12.

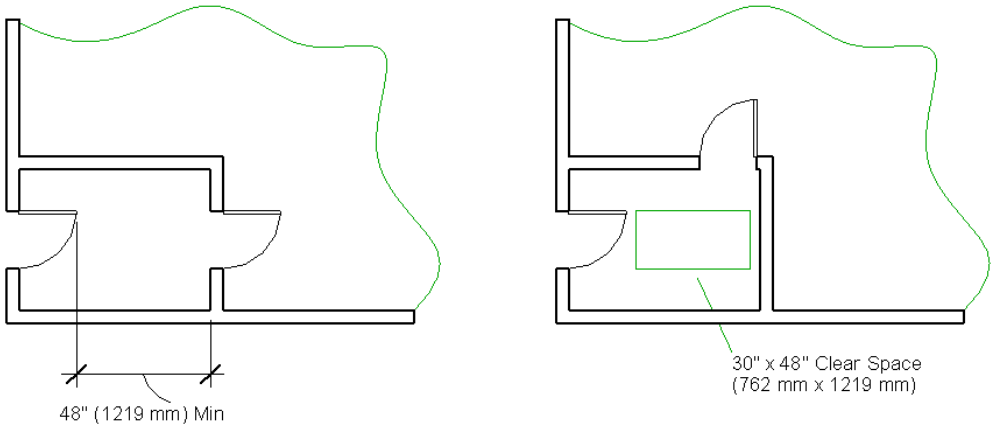


Figure 10 - Diagrams illustration of doors in series for IBC 1008.1.8

** Illustration is reproduced based on IBC 2003 Commentary [70]*

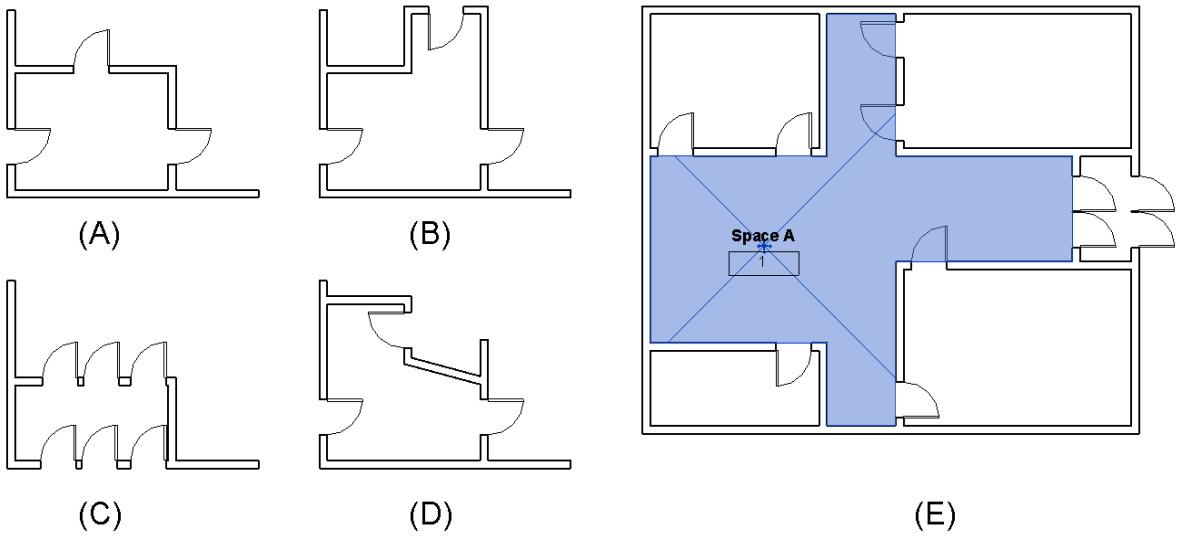


Figure 11 - Several possibilities that should be considered when evaluating a space for doors in series. In (E) the focus is on Space A.

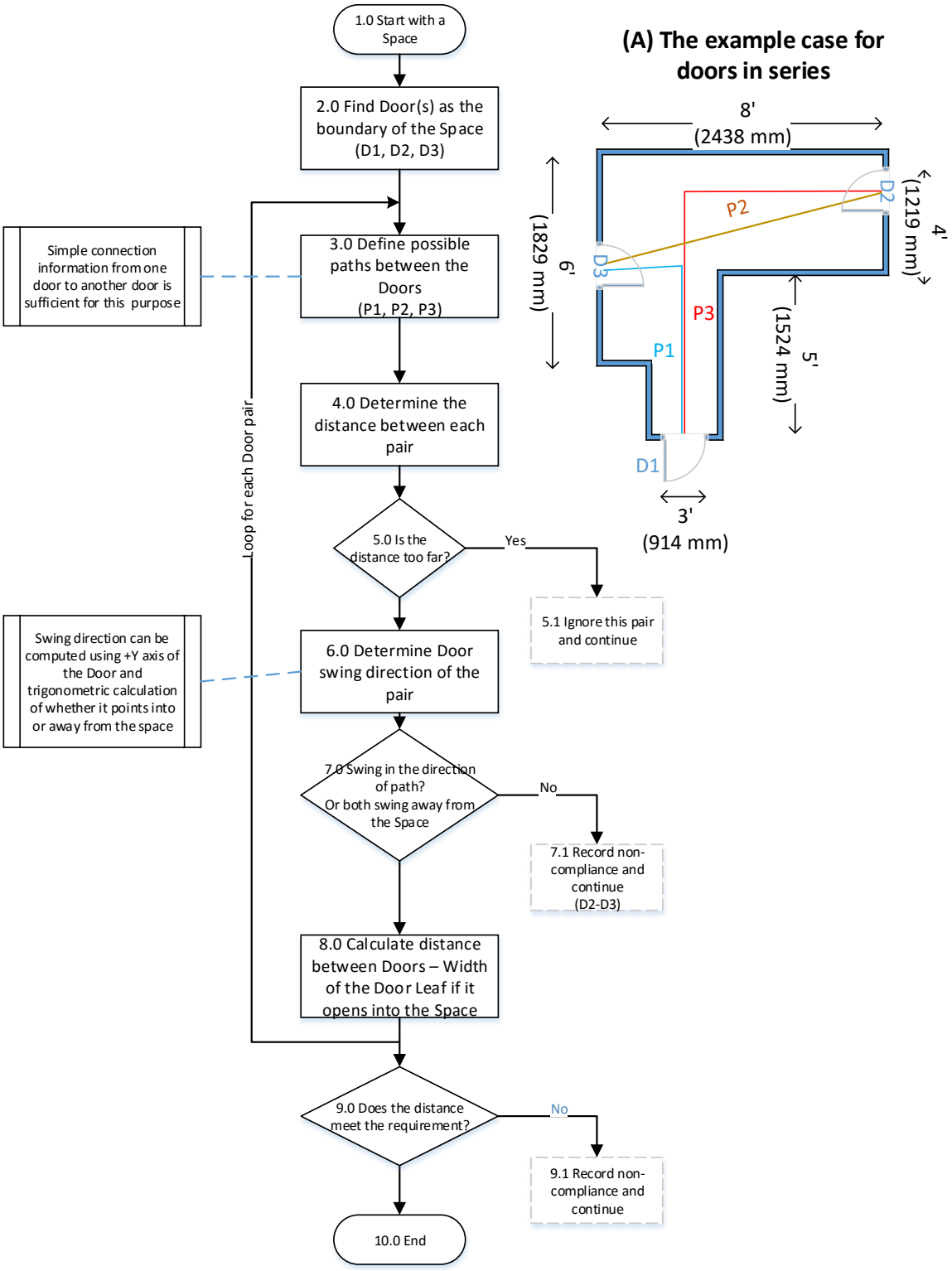


Figure 12 - Checking Process for IBC 1008.1.8

The example above shows the generic algorithm that is applied to a particular case such as shown in Figure 12 (A). It can be applied to the cases illustrated in Figure 11 since it deals with these key checking items:

- Determining candidate doors in series by pairing all doors adjacent to the space being checked (steps 2.0 and 3.0). These steps will require trigonometric calculations to find the path and to determine whether a straight line distance of 48” (1219 mm) is sufficient or the need of defining a box of 30”x48” (762 mm x 1219 mm) is required (Figure 10).
- Eliminating pairs that cannot be defined as doors in series, i.e. they are too separated and do not serve the purpose of doors in series (steps 4.0 and 5.0). This step will eliminate a case illustrated in Figure 11 (E) and other spaces with similar conditions.
- Checking sub-rule no. 1 requires doors to be either opening in the same direction or both opening away from the Space (steps 6.0 and 7.0). This can be achieved using the accepted convention of +Y-axis direction as the door swing direction and simple trigonometric calculation to determine that the point in +Y-axis is inside or outside of the Space.
- Checking sub-rule no. 2 that requires that any doors in series must have a minimum gap of 48” (1219 mm) between the doors discounting distance taken by the door leaf that opens into the Space or there is a clear area in a box of 30”x48” (762 mm x 1219 mm) in front of the door (steps 8.0 and 9.0), making use of information obtained in the steps 2.0 and 3.0 above. Additional information that is often inferred and necessary to limit the scope of search for the solution space is the maximum distance between doors that can be reasonably considered as candidates for the doors in series. In this example 9’-10’ (3 m) is deemed to be a reasonable distance that sufficiently cover the rule requirements and to filter out unlikely candidates from the search space.

In the specific example selected (Figure 12 (A)), the rule starts with an evaluation of the candidate Space (step 1.0). The Space returns three doors (D1, D2, D3) as its boundaries in step 2.0. With three doors, 3 pairs of doors and 2 directions need to be evaluated (step 3.0). More detailed descriptions on this combinatorial issue will be discussed in the next section. Since there is no door pair that has a distance longer than specified, no door-door path is eliminated (Steps 4.0 and 5.0). Using the sub-rule no. 1, i.e. Doors in series must swing in the same direction or both swing away from the Space (steps 6.0 and 7.0). P2 connecting D2 and D3 will fail this test for both directions. P1 and P3 will pass both sub-rule no. 1 and sub-rule no. 2 that specifies the distance between the two doors minus the leaf door that opens into the Space (steps 8.0 and 9.0).

3.7.2 Example of combinatorial issue

The example of the door in series above (Figure 12(A)) illustrates the need for a rule checking system to deal with combinatorial issues. The space being checked offers six combinatorial configurations to be evaluated and there can be any number of the possible configurations that meet or fail the requirement. Table 2 below describes the six conditions:

Table 2 - Six Combinatorial Cases in Rule Check for IBC 1008.1.8.

Case	Potential Path	Door Swing Direction	Condition to satisfy [Swing (D_i) = S && Swing (D_j) = S'] or [Swing (D_i) = S' && Swing (D_j) = S']
1	P_1 (D_1, D_3)	Swing (D_1) = S' Swing (D_3) = S	Pass
2	P_1 (D_3, D_1)	Swing (D_3) = S Swing (D_1) = S'	Pass
3	P_3 (D_1, D_2)	Swing (D_1) = S' Swing (D_2) = S	Pass
4	P_3 (D_2, D_1)	Swing (D_2) = S Swing (D_1) = S'	Pass
5	P_2 (D_2, D_3)	Swing (D_3) = S Swing (D_2) = S	Fail
6	P_2 (D_3, D_2)	Swing (D_3) = S Swing (D_2) = S	Fail

Where

$P_i(D_i, D_j)$ Path i connecting Doors D_i and D_j in the direction of $D_i \rightarrow D_j$

$Swing(D_i) = S\ or\ S'$ Swing direction of Door I, either into Space (S) or away from Space (S')

Depending on the exact rule that is being applied, the combination may be reduced. The above rule (IBC 1008.1.8) is in the context of Path of Egress. In this case only one path direction is relevant depending on the direction of the Egress, which leaves only cases 1, 3, 5 or 2, 4, 6 to be relevant. While the combination is reduced in this case, the result is unaffected, i.e. Path 2 (P_2 between D_2 and D_3) fails the checking criteria.

3.8 Class 3 - Rules that require extended data structure

This class of rules requires an extension to the data structure that encapsulates higher level semantic conditions of building data. The main idea is to be able to "compute once, use many" since such information typically requires extensive computation often involving geometry operations. Multiple data structure construction strategies may apply. In this case this thesis offers one of the possibilities. Typical use of this class of rules is building code checking that involves complex requirements.

3.8.1 Degree of complexity/required tool

To be able to solve rules in this class, software relying on geometrical, topological and other properties and algorithms is often required (Figure 13). Spatial relationships may be important. To effectively deal with complex geometrical and spatial operations, a Solid modelling library may be required to perform complex geometric operations. Derivation of topological graphs may be required as well and algorithms such as shortest path may be involved. The most mature implementation that does these is FORNAX™, which is the base implementation of CORENET ePlanCheck [2, 43, 44].

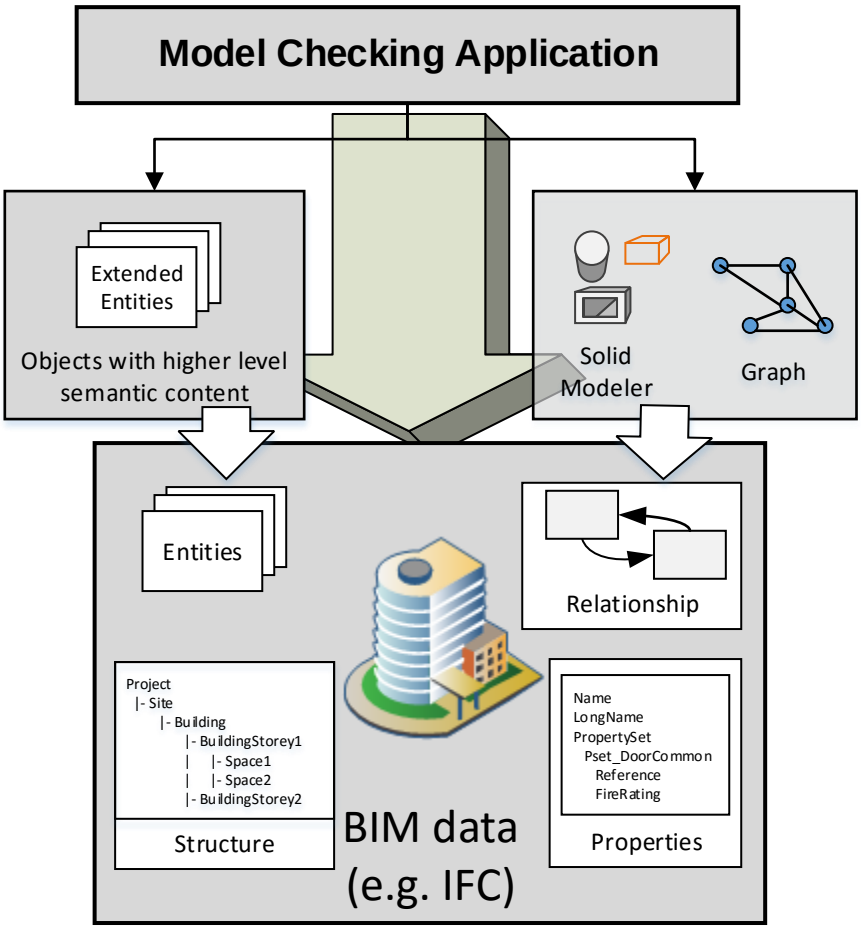


Figure 13 - Diagram for Typical Application Implementing Class-3 rules

FORNAX™ extends IFC model data with a new structure that defines higher level abstractions of the building model [44, 45]. One example implementation described is:

- *Singapore Fire Safety and Shelter Department Fire Code 2.3.5 (as implemented in CORENET IBP)*
  Basement Exit Staircase
  a) Any exit staircase which serves a basement storey of a building shall comply with all the applicable provisions for exit staircase, and
  b) Such exit staircase shall not be made continuous with any other exit staircase which serves a non-basement storey of the building, and
  c) Basement exit staircases which are vertically aligned with the exit staircases of non-basement storeys shall be separated from such other exit staircases by construction having fire resistance for a minimum period equal to that required for the enclosure.

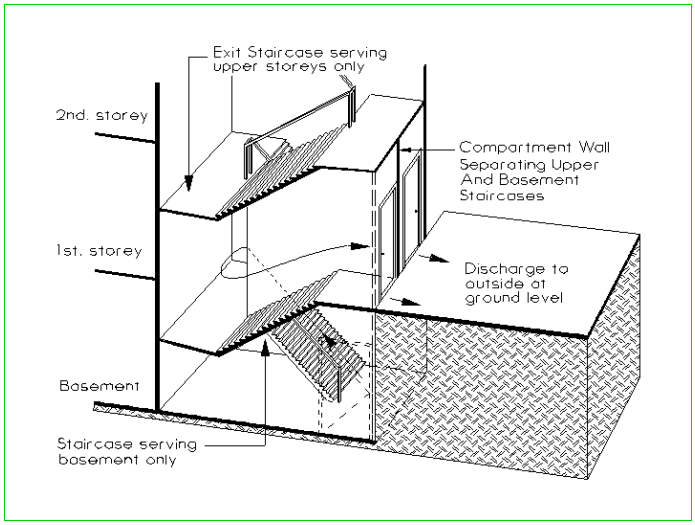


Figure 14 - Illustration for IBP Fire Code Clause 2.3.5 (a, b and c)

This rule requires new concepts to be introduced:

- Vertical Shaft, that will be useful not only to connect exit staircase spaces to form an exit path through the staircase, but is also useful for other fire safety requirements, e.g. fire rating of the shaft or checking requirement for pressurization, which requires the compartment to be airtight.
- Discharge, which is access that is directly opened to the ground level oriented towards the outside of the building.
- Separation of the discharge space from the upper level and the basement if it is modelled as one space separated by other objects such as railings.

Here is another example in this category:

- IBS CP10 Installation and Servicing of Electrical Fire Alarm Systems, Clause 4.3 Spacing and Locations of detector

Clause 4.3.3
Spacing between detectors.

1. The distance between detectors for flat ceiling shall not exceed:
a) 7m for areas other than corridors and
b) 10m for corridors (See Figure 15)

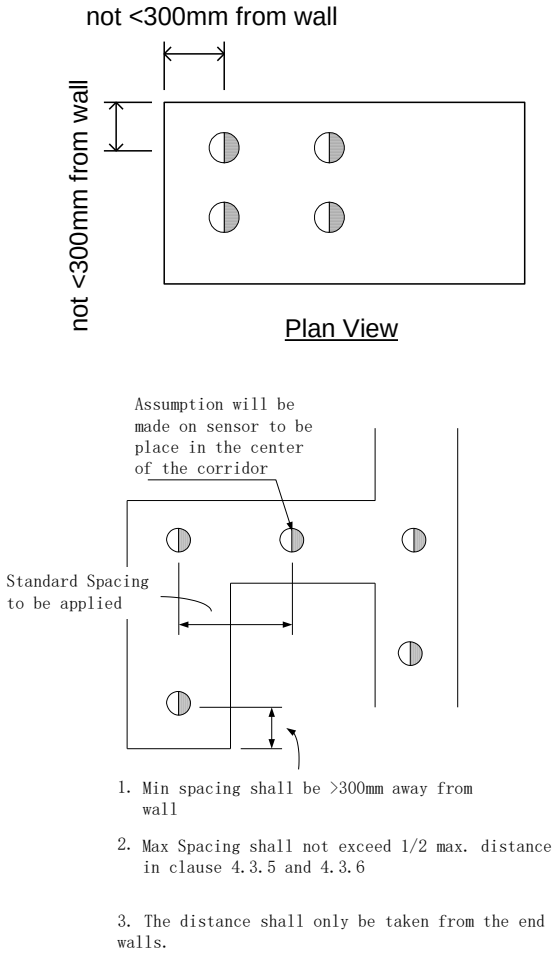


Figure 15 - Illustration for IBS CP10 Clause 4.3.3

This rule requires a sophisticated geometry engine to create a coverage map using distance triangulation between detectors [71]. This allows identification of the nearest neighbor detectors and hence the distance between the detectors can be checked (Figure 16). This example shows the need to “compute once use many” for generating the triangulated network, in that it can be re-used in many other rules related to the placements of detectors. The same technique is also required in other similar clauses applicable to sprinkler systems.

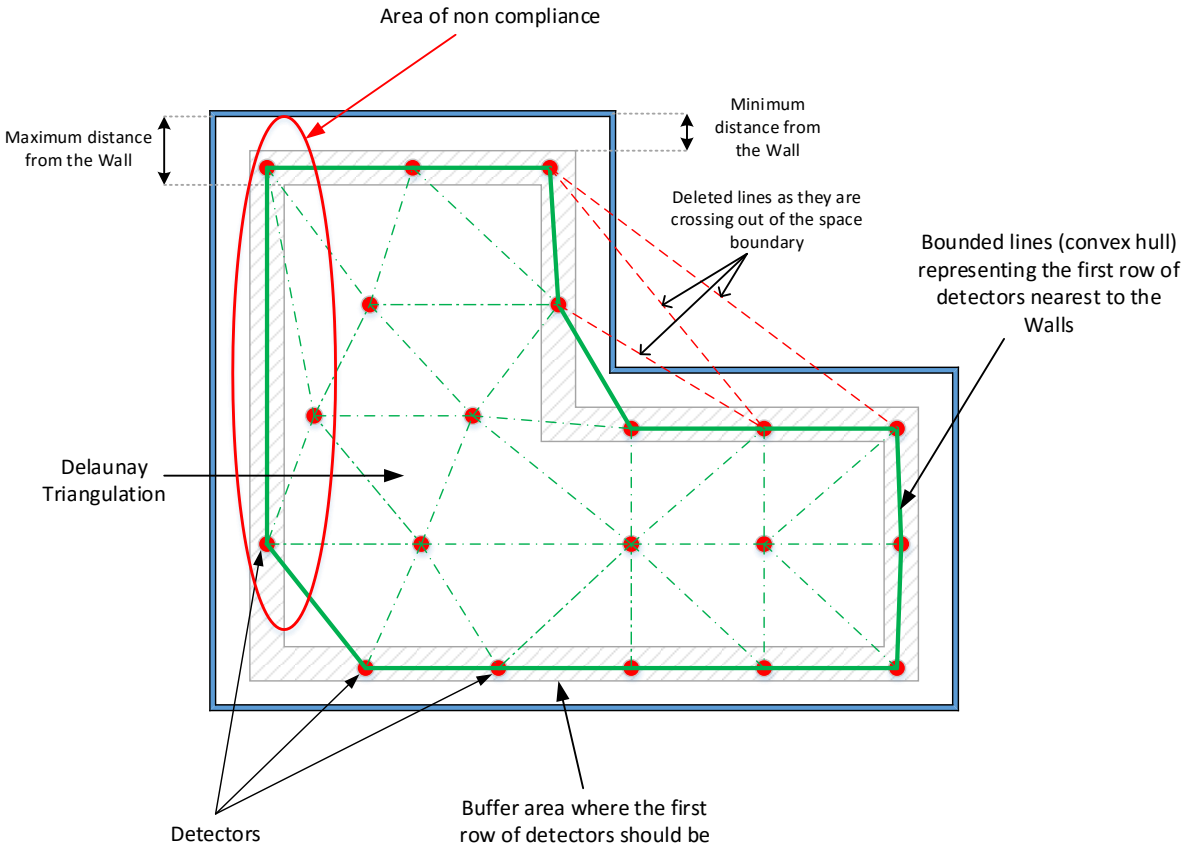


Figure 16 - Coverage Map of Detectors for Detection of Distances and Identifying the First Row

3.9 Class 4 - Rules that require a "proof of solution"

This class of rules is a collection of rules that do not strictly ask for compliance or non-compliance, but rather requires a "proof of solution". The rules usually focus more on how the building model proves compliance rather than merely fulfilling prescribed criteria. It generally represents performance based codes or other similar rules. However, it can also be applied to rules that can be modified to satisfy the compliance by looking into a solution in the form of additional model data being inserted into the existing one, usually either temporarily or virtually. Generally the application of this class of rules is more interested in the solution, which may have more than one acceptable answer, all of

which eventually can be traced to the final answer of whether a design (or the design with additional information added into the design) complies with what the rules expect. The source of the solution is typically a knowledge-based facility that is coded into the system [27, 72] (Figure 17). Such a knowledge base is typically captured and continuously updated as new knowledge arises. For example in the case of designs that affects fatality in a construction site, there are design rules such as "Design columns with holes at 21 and 42" (0.53 – 1.06m) above the floor level to provide support locations for lifelines and guardrails". This rule should be added into the knowledge base as the result of statistics that there were a relatively high number of fatalities caused by this reason [73]. In recent years, with the increasing number of "green" buildings, new types of installation such as photovoltaic (PV) panels on the roof surfaces increases the risk of falling objects and therefore would require a new rule to define protection around the edges and distance of such installation to the edges [74]. This new knowledge should be easily added into the existing system and automatically extend the coverage of an existing rule checking capability that deals with protection from falling.

3.9.1 Degree of complexity/required tool

It is expected that the degree of complexity does not significantly increase from the other classes, but it has new requirements in order to capture knowledge and present it as a viable solution. This enters the realm of expert systems, though it probably focuses more on domain specific knowledge [75]. Many of the applications currently seem to be suitable for the construction domain where there are many temporal conditions that highlight the need for solutions, such as those related to temporary structures (shoring and temporary structural supports), and also those related to safety issues.

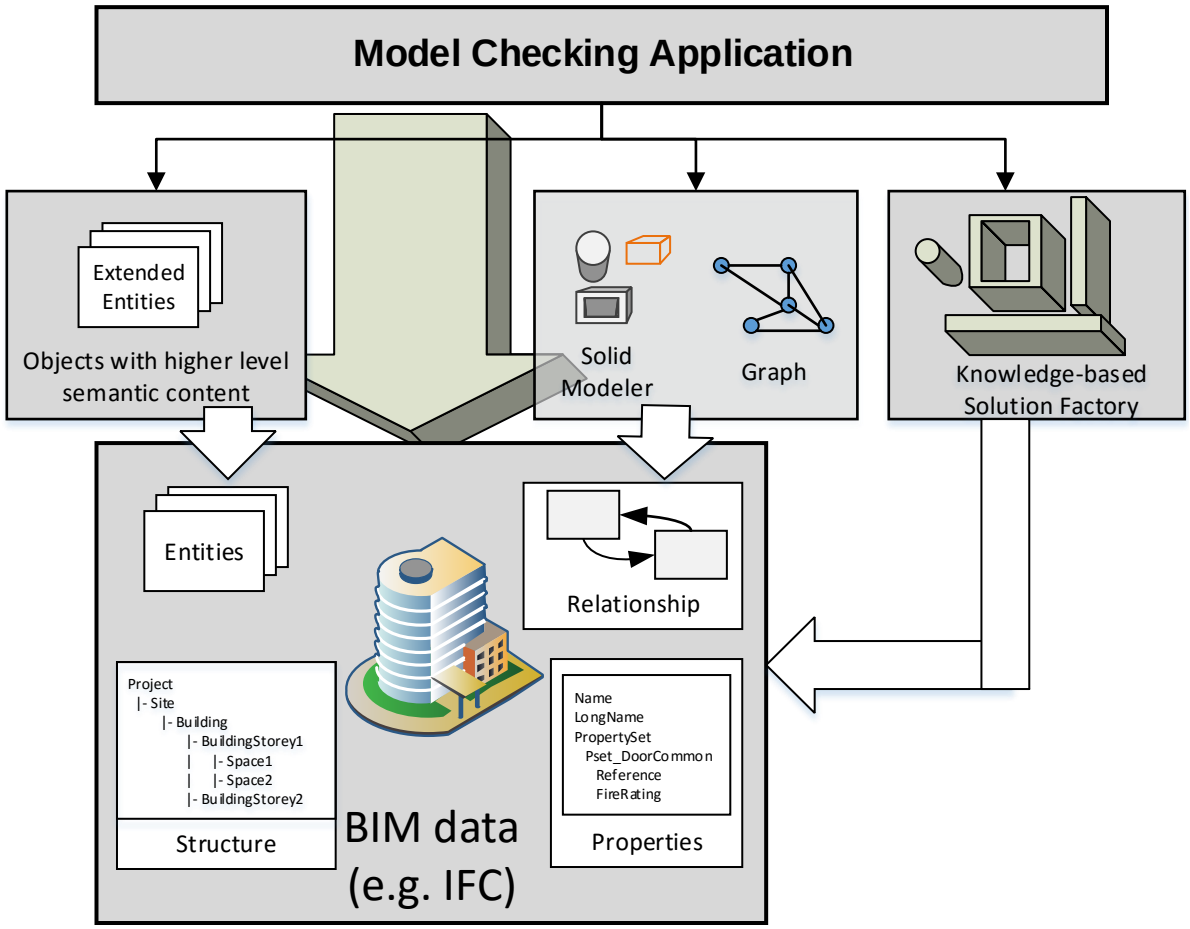


Figure 17 - Diagram for Typical Application Implementing Class-4 Rules

This is a new area of rule checking. There are not many examples available currently. The work on safety from falling done by Zhang serves as a good example of an implementation for this class of rules [27].

There is also a possibility that rules that traditionally fall into other classes, especially Class-3, will evolve into Class-4 when one looks at the problem in a different perspective. For instance, the example presented in Class-3 (IBS CP10 4.3.3) can be restated as a rule. If this is so, then the solution would be an input that the designer could use to correct his design when non-compliance is detected in terms of the correct placement of the detectors. In this case an algorithm can be developed using the same

concept of triangulation to assist the designer to highlight suggested placements of the detectors that will satisfy the rule by optimizing the distance between the detectors and making sure the first row fits into the buffer space as highlighted in Figure 16.

Table 3 - Example of rules that fit the four rule classification

Rule classification	Source	Rule description
Class-1 Checks based on a single or small number of explicit data	Solibri Model Checker (SMC) rules [15]	General BIM file structure rules *"Model Should Have Components"*, checks: • Explicit entity types • Element's attribute describing construction types, and • Assigned classification name (using IfcClassificationReference) Spatial information rules: *"Spaces Must Be from Agreed List"*, checks: • IfcSpace.Name (space number), LongName (space name), Type (classification)
	International Code Council IBC 2009 [67]	*"909.20.4 Mechanical ventilation alternative. The provisions of Sections 909.20.4.1 through 909.20.4.4 shall apply to ventilation of smokeproof enclosures by mechanical means.* *909.20.4.1 Vestibule doors. The door assembly from the building into the vestibule shall be a fire door assembly complying with Section 715.4.3. The door assembly from the vestibule to the stairway shall not have less than a 20-minute fire protection rating and meet the requirements for a smoke door assembly in accordance with Section 715.4.3. The door shall be installed in accordance with NFPA 105."* Checks: • Pset_DoorCommon.FireRating *"1008.1.2 Door swing. Egress doors shall be of the pivoted or side-hinged swinging type."* Checks: • IfcDoorStyle.OperationType for value of the SWING type (Single or Double Swing)
	Singapore Fire Code (prescriptive) Chapter 3 clause 3.2.5.e [76]	*"The fire command centre shall be separated from other parts of the same building by compartment walls and floors having fire resistance of at least 2 hours.",* checks: • All space boundary elements (through IfcRelSpaceBoundary) and their Pset_<object>Common.FireRating for value at least 2 hours.

	MVD checking	Checks: • Existence of valid entities (IfcObject and its subtypes) • Valid relationship between entities (IfcRelationship and its subtypes) • Their attributes and propertysets and possibly their valid values.
Class 2 Checks based on simple derived Attribute Values	SMC rules [15]	Component Properties rules: *"Free Area in Front of Components Rule"*, checks: • There is no obstruction in front of an object like Windows, Doors or specified furniture (It can also check both sides of the object) • It requires extra information processing what objects could be in front or behind Door or Window. *"Components Must Touch Other Components"*, checks: • Object spatial connectivity by face coincidence other defined objects and how much the area intersection and checks if object has the acceptable height.
	Construction coordination	Clash detection • It requires an object to be aware of any other objects around it.
	Singapore Code of Practice on Sewerage and Sanitary Works; Part 3.2 Sanitary Plumbing Systems [68]	*(42) 3.2.2 Design Criteria* *f) The discharge pipe shall not be located in places where it can cause health and safety hazards such as locating the discharge pipe above any portable water storage tank and electrical transformer/ switchgear.* Checks: • Discharge Pipe needs to be aware of specific objects underneath it as long as there is no protection.
	Hospital design	*"All patient rooms must be visible from the nurse station"*, checks: • Simple geometry computation that evaluate "line of sight" from the nurse station to entrances for each of patient rooms under supervision of the nurse station.
Class 3 Checks based on extended data structure	Construction best practice	*"There has to be a direct access to MEP control devices that are concealed for the purpose of maintenance through access opening of minimum size of 18"x24" (457 mm x 610 mm). Where there is no direct access from the opening, it is permitted to have indirect access through crawl space that has minimum clear dimension of 22" x 30" (559 mm x 762 mm) at the cross section uniformly applied throughout the entire crawl space"* • Requires spatial information of the concealed space, creation of sweep solid to create a required crawl space starting from the nearest access opening to the front of the control device.

	GSA Courthouse Design – Circulation requirements [77]	"*The USDC courtroom should be accessible from the Prisoner HLDG.CELL only through secure circulation (3-14-16)*" "*The judge's chambers are accessed from restricted circulation with convenient access to the courtrooms"* • These requirements imply the need to have space connection graph to allow evaluation whether a space is accessible from another space though one or multiple intermediate spaces such as circulation spaces. All traversed spaces must be of type 'secure'.
	Singapore Building Control Act (Chapter 29) 1990, Division 5 Staircase [68]	*Regulation 44 (1) Protection of staircase and staircase landing* *Every staircase or staircase landing shall be protected on any side overlooking an air-well, courtyard, void or external open space by either a railing, parapet or balustrade capable of resisting the lateral loading as specified in the Table 4 of the fourth Schedule*, checks: • Side edges of the stair, the number of steps and evaluating existence of protection as well as the depth of fall from the edges Involves implicit requirements: • The protection may start only when the staircase has more than 5 steps • The danger of falling is defined to be an open edge that has a drop more than 1 meter high
	ICC IBC 2009 [67]	*1005.2 Door encroachment. Doors, when fully opened, and handrails shall not reduce the required means of egress width by more than 7 inches (178 mm). Doors in any position shall not reduce the required width by more than one-half. Other nonstructural projections such as trim and similar decorative features shall be permitted to project into the required width a maximum of 1-1/2 inches (38 mm) on each side.* • This rule becomes more complex when one considers in the real life scenarios where: o Egress path is not simple and straight o There are more than one door opening into the Egress path
Class 4 Checks and suggests corrective actions or solutions	Occupational Safety and Health Administration (OSHA) Protection from falling in Construction [78]	*OSHA health and safety rule for protection from falling during construction*
	Construction and Operation & Maintenance related best practice	*"Ensure a footprint of free space for placing a ladder to reach access opening on the concealed ceiling, or MEP control devices"* • While this rule may simply return compliance or non-compliance of access opening or MEP device that do not have enough footprint of free space underneath, it is a

		good candidate for highlighting how much space it requires to fulfil the requirement. It will help designers to alter the model to satisfy this requirement during the pre-construction coordination process. One possible solution is placing a ladder object underneath the access opening or MEP control device and highlight those either there is no enough free space or the item to access is beyond the reach by standard ladder. *"Find possible accessible path to move a large equipment into a building under construction"*, or similar requirement for Plant: *"Find possible path and placement of crane to lift a plant equipment out of its current position for removal or replacement of the equipment"* • This requirement involves analysis of the building or plant layout to identify the possible paths and clear space for moving the object into the final position or out from its current position. It involves path generation, accessibility analysis on the path possibly using swept solid along the path, n-shortest paths algorithm, and clash detection.

Table 4 - List of research work and commercial applications implementing different classes of rules

Class 1	Class 2	Class 3	Class 4
Checks based on explicit data	Checks based on simple derived Attribute Values	Checks based on extended data structure	Checks and suggests corrective actions or solutions
Solibri Model Checker [15]	Solibri Model Checker [15]	GA Tech GSA courthouse design circulation check [32], which is based on Solibri Model Checker	GA Tech's Automatic Safety Checking of Construction Models and Schedule [27]
Automated MVD checks: - IFC2x3 CV2.0 certification by BuildingSMART and iabi [79] - Digital Alchemy, that is used in GSA CD BIM 2010 certification [80]	Applications with Clash detection capability: - Navisworks [81] - Tekla BIMsight [82]	FORNAX™ as implemented in CORENET ePlanCheck [44]	Various proprietary domain specific implementations*, e.g.: - Crane Simulation system by JGC using Navisworks - Autodesk's simulation for Lockheed Martin on risk assessment of moving large objects inside a building based on 3D Studio Max
Revit Model Review (subscription add-in) [83]	FORNAX™ that is used as an engine for CORENET ePlanCheck implementation [43]		
Navisworks using search set# [84]			
Research project by QUT, Australia [20]			
EDM rule schema based: - Norway pilot project [33] - Xabio [85]			

** No public information available, they are based on the author's involvements with the developers of the applications.*

 # IFC support in Navisworks for search set will be limited to Entities and Properties

CHAPTER 4

ANALYSIS OF THE LOGIC STRUCTURE OF BUILDING RULES USING CONCEPTUAL GRAPH[2]

4.1 Introduction

One of the important steps in the rule checking process is rule interpretation [6]. Experience in implementing CORENET ePlanCheck in Singapore shows that the interpretation step can take as much as 30% of the total time to implement a rule. Complex rules typically found in building codes are a combination of several aspects that contribute to their complexity, i.e. the language structure, the domain knowledge embedded in the rules that includes hidden assumptions, and their logic structure. Added to these technical aspects of the rules is the human aspect of the interpretation. A study by Fiatech confirmed that when human interpretation is involved, inconsistencies are expected. Different officers tend to interpret the rules differently, often colored by their experience and locality [86]. Some rules in CORENET ePlanCheck implementation went through multiple iterations and revisions because of the same reason when multiple reviewers were involved. It does not help that in a typical development, software developers are not the ones directly involved in the rule interpretation.

Rule checking does not stop at the development effort. It also involves the second workflow after the rules have been implemented. Different sets of users are now involved in the process that involves a data exchange, which needs to be consistent with the rule

---

[2] *The content of this chapter has been published in the International Conference CIB W78, Eindhoven, the Netherland, Oct 2015. This chapter is the extended version of the paper*

implementation. This requires a different form of the current knowledge transfer to the modelers. This aspect of AEC specific workflow has been well defined and a standard process has been proposed and used within the BuildingSMART community using IDM (Information Delivery Manual) and MVD (Model View Definition). The challenge in rule checking implementation is to integrate the knowledge transfer during the interpretation, implementation process and the actual usage of the rule. This should be done with minimum information loss (Figure 18). Good communication and documentation becomes critical in this process. Unfortunately without a systematic methodology it has proven to be a significant challenge. Experience in implementing an automatic rule checking system in CORENET ePlanCheck shows that the voluminous documentations did not reduce the issue of knowledge loss along the process, and in some cases it added into the problem.

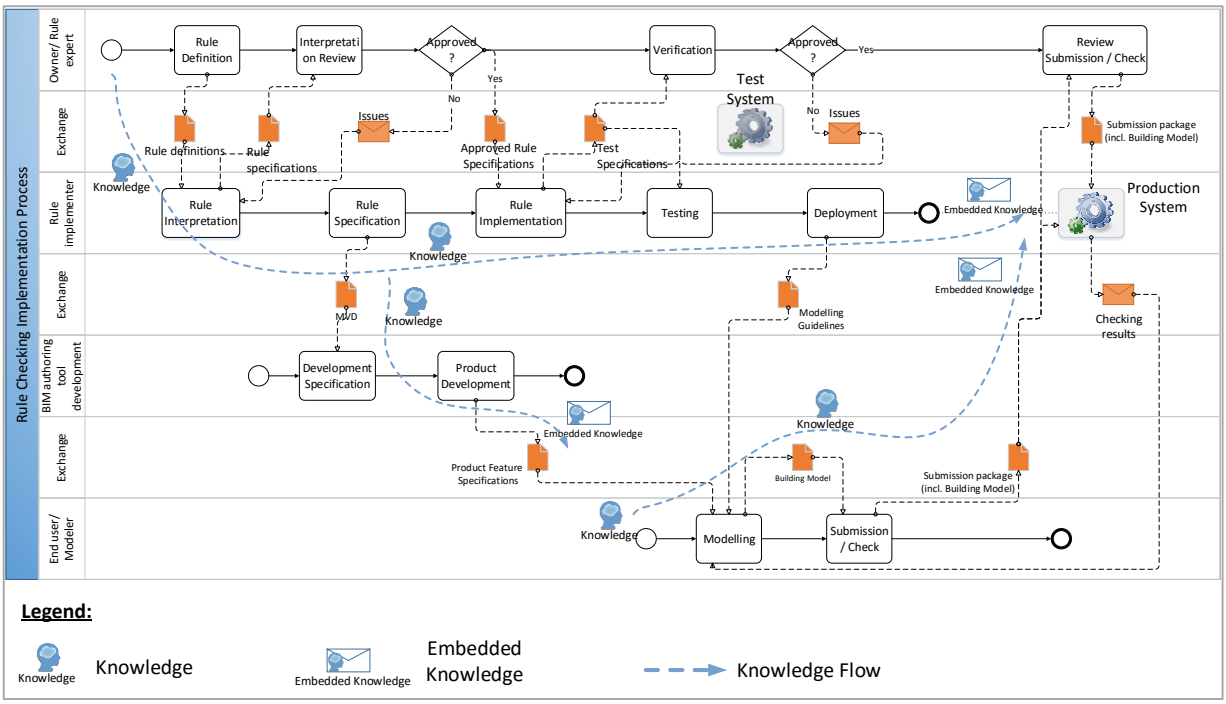


Figure 18 - Typical Rule Implementation Process and the Knowledge Flow

Recognizing the importance to address the knowledge gap and reduce the information loss, this study proposes the use of the Conceptual Graph (CG) to capture the

rule requirements in form of semantic knowledge representation. The CG is designed to capture knowledge of the rule into its basic logic structure and the data involved. The aim is to enable effective communication between all users and to identify exact data, relationships between data, and any required functions to encapsulate the complex algorithms involved in solving a rule. One important feature of using the CG for this purpose is a built-in mechanism to capture the requirement for derived data and relationships that are vital to rule checking implementation.

4.2 Semantic Knowledge Representation of Building Rules

In recent years, several approaches have been proposed to capture building related rules into certain forms of knowledge representations. For example, El-Gohary et al proposed the use of Natural Language Processing (NLP) to "interpret" building codes [11, 13]. Hjelseth and Beach used a methodology called RASE (Requirement, Applicability, Selection, Exception) to tag the building codes into four categories of ideas to drive the computer implementation of automated rule checking [10, 22]. Both approaches focus on the structure of the rules and have not sufficiently addressed the semantic-interpretation issue of the rules and whether the building representation is sufficient to support the richness of language expression written in the rules. In this research, the rules are viewed as knowledge assets complete with their association with the building representation that they mean to check. As previously mentioned, in this research, only the widely used open standard IFC (Industry Foundation Classes) by BuildingSMART is used. IFC is now an ISO standard [87] and is supported by all major BIM authoring tools.

The first task needed is to choose a suitable representation to capture semantic knowledge of the rules. The most suitable method appears to be from the field of Knowledge Base (KB) that is a branch of Artificial Intelligence (AI). But even within KB, there are many different approaches including First Order Logic (FOL), Description

Logic (DL), and Conceptual Graph (CG). CG was originally proposed by Sowa in 1976 [88] and further developed in 1984 [89]. The Conceptual Graph (CG) offers intuitive, easy to read and suitable to capture semantic knowledge representation of the rules [90]. CG has its semantic foundation in FOL and the basic form of CG can be mapped 1-to-1 directly to FOL.

Thus the goals in using the CG are:

- As an expressive tool to capture knowledge of rules in terms of their requirements for automation that are easily understood by the rule experts, who are typically not familiar with computer programming.
- Ability to capture data requirements of the building objects and their relationships or interactions with other building objects, including constraints.
- Direct mapping of the CG concepts into IFC entities, derived entities and extension functions. The mapping is important for both defining MVDs and for software development efforts.
- Ability to breakdown complex rules into their atomic rules in a systematic and standardized way.

4.2.1 Conceptual Graph as a Knowledge Representation of Building Rules

With its history in semantic networks, CG defines rectangles to represent concept nodes, ovals represent conceptual relations, and diamond shapes represent functions (an extension to CG). The nodes are connected by arcs with an arrowhead pointing to the ellipse, which marks the node as the first argument of the relation. The node with the arrow pointing away from the ellipse marks the last argument (Figure 19).

A Concept node typically represents an object, but it can also be extended to represent a whole atomic rule. This is achieved using a concept called coreference that

link two different nodes. Coreferent nodes should be able to be merged into just a single node. They are represented with dashed line (Figure 20).

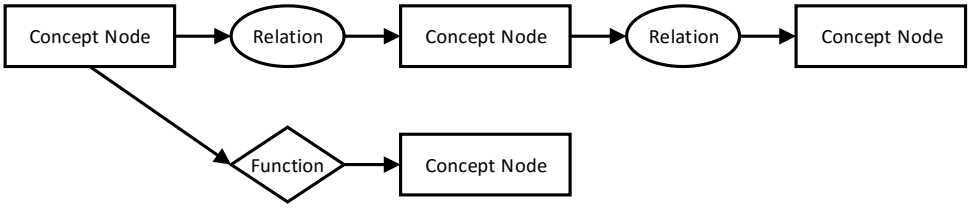


Figure 19 - Basic Definitions of Conceptual Graph

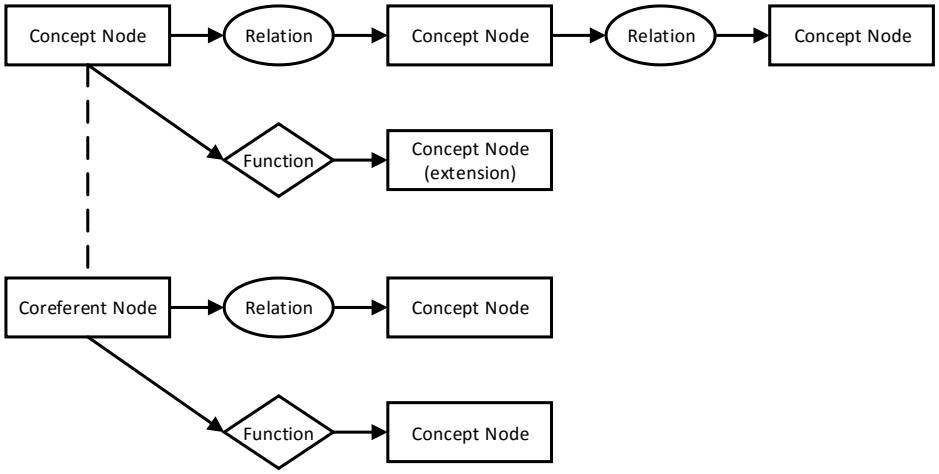


Figure 20 - Coreferent Node

The CG is used to represent the semantic knowledge of the rules. A rule will be represented by a series of connected graphs as above. Additional notation used for the graph is for Constraint. Constraint is described in a similar manner as a rule, except it uses a different style, i.e. shaded. Constraint can be applied to any node in the graph, which indicates to which concept the constraint applies (Figure 21). Since constraints defined specificity of the concepts they apply to, they must rely on the explicitly defined properties or relationships in the IFC file. They include element and type property names, property values, classifications, or relationships.

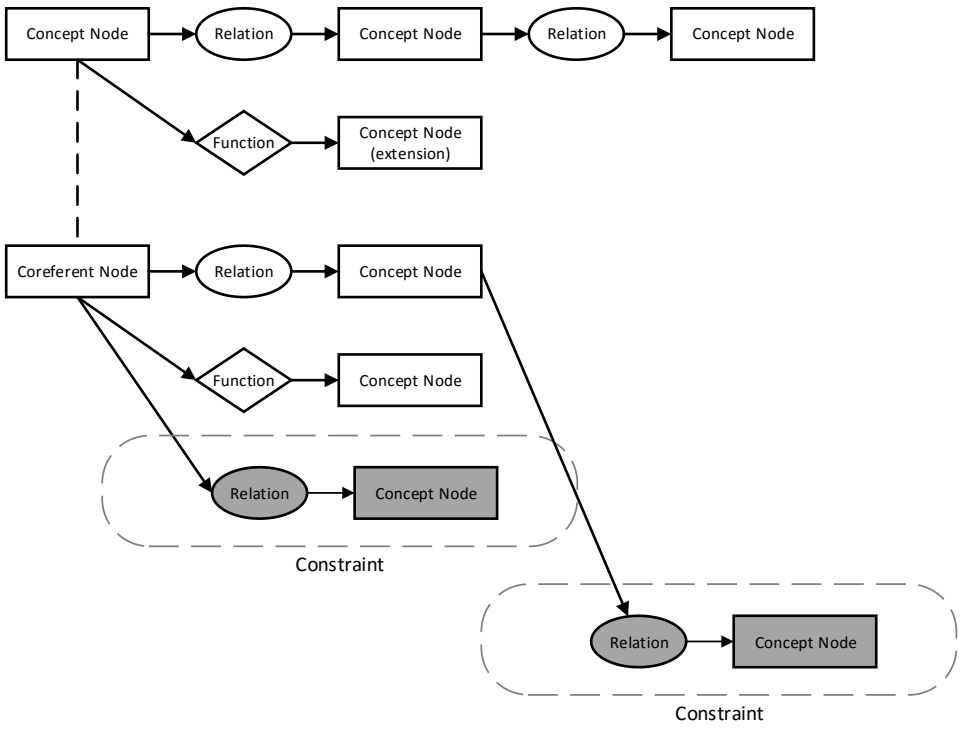


Figure 21 - CG with Constraints

4.2.2 Notational Extension

Due to the complexity of building rules, there is a necessity to add a few notational extensions into the graph for improved readability of the CG. Figure 22 shows such extensions:

- Nodes with dashed line borders represent a derived concept or concept that will require additional support during the implementation using computer algorithm. The requirement for a derived concept has been identified to address more complex classes of rules [91]. The derived concept generally requires support from a computer algorithm that is applied to the basic building model. FORNAX^{TM} , which was developed for CORENET ePlanCheck, used the same concept [44], and very recently a rule checking effort in the UK took a similar approach [92].

- Specific labels: OR and (NOT) ¬ to represent logical disjunction (∨) and negation (¬). By default, if there is more than one link connecting a node, the operation is a logical conjunction (∧). A thin line box surrounding the negation block is part of the standard CG.
- Thin dashed line with rounded rectangle represents a special block, which could be used to show the constraint block(s) or the exception rule.
- A dotted line connecting a concept node to a double border box indicates a dependency for the concept that is specified in another rule. This is important information that connects a certain concept with a specific rule.

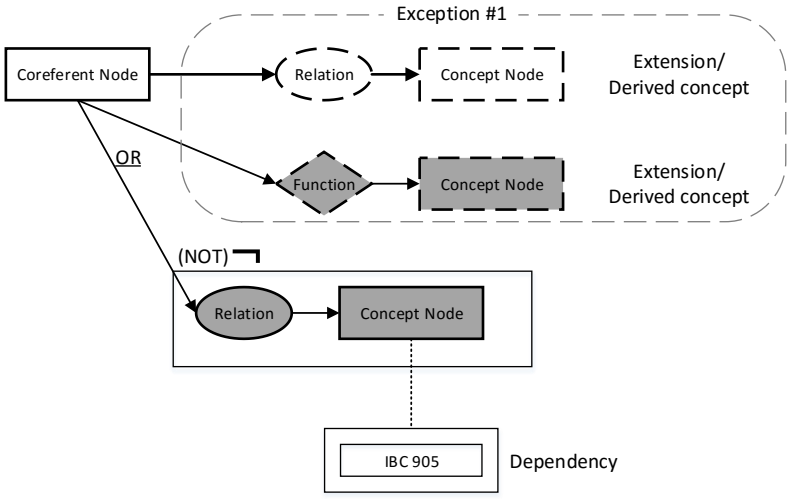


Figure 22 - Notational Extensions for CG

4.3 Translation of Rules into the Conceptual Graph

Translation of rules into CG is not always straightforward sequential mapping because of the way the rules are written. The following steps are used in performing the translation, which is generally part of the interpretation step:

1) Identify the main concept that the rule is applicable to. This step should identify a concept without any qualification, as the minimum. Most of the time it will have some kind of filter or specifications on what type of specific concept it is applied to. For example:

- "*Spaces (instance) must be from agreed list*" [15] will identify the concept applicable to this rule as a "Space" instance without any further specification. In First Order Logic (FOL) this rule is represented with $\forall x(Space(x))$
- "*The underground building shall be equipped throughout with a standpipe system …*" (IBC 2009 405.10) will identify that the concept is applicable to a building, but not just any building. It specifies an underground building. In FOL it is represented with $\exists x(Building(x), Property(x, type{:}UNDERGROUND))$

In some cases the concept is not very explicit. In this case a level of abduction is needed. For example:

- "*Model should have components*" [15]. In this rule, model refers to the building model as a whole because the rule requires any type of entity that can be specified. Since the rule comes from Solibri Model Checker, which provides a template, this rule can only be operational once the user assigns what entity type(s) the rule should apply to.

2) Identify atomic sub-rule(s). A rule, especially in building codes, often specifies more than one sub-rules that are relatively independent, except that they are operating on the same entity. In this case, the sub-rules will be defined as separate rules under the same heading. For example:
   - "*Doors, when fully opened, and handrails shall not reduce the required means of egress width by more than 7 inches (178 mm). Doors in any position shall not reduce the required width by more than one-half. Other nonstructural projections such as trim and similar decorative features shall be permitted to project into the required width a maximum of 11/2 inches (38 mm) on each side.*" [67].

This rule is applicable to doors that open to the egress path. There are three sub-rules, one that deals with the space occupied by the door at the fully open position, one that deals with the reduction of egress width due to the door opening, and one that specifies the maximum projection of the trim and decorative features of the door into the egress space.

3) Identify atomic constraint(s). The general structure of a building rule is a specification of the main building entity with its details, followed by one or many constraints. The constraints are not restricted to the main entity, but can also apply to other entities that are related to the main entity or even to an entity within the constraint, i.e. constraint within constraint. This increases the complexity of the rule. For example:
   - "*All patient rooms must be visible from the nurse station*"

   This rule has a constraint on the main entity, the nurse station. The constraint specifies that the patient rooms must be within the line of sight from the nurse station.
4) Define the appropriate CG of the rule by connecting the concepts using relations and functions until a consistent semantic is clearly self-describing.

4.3.1 Applying the Semantic Representation CG to Building Rules

In this section, several rules are selected to represent the ranges of rules that are applicable to buildings. Several rules are selected from different classifications of rules as defined in [91], mainly for class-1 to class-3. Class-4 does not create new semantics or complexity, but it introduces requirements in terms of an algorithmic solution to present a "proof of solution", and therefore it does not require additional semantics than the other three classes.

4.3.1.1 Class-1 rule (rules that require a single or small number of explicit data) example

"Spaces must be from agreed list" [15]

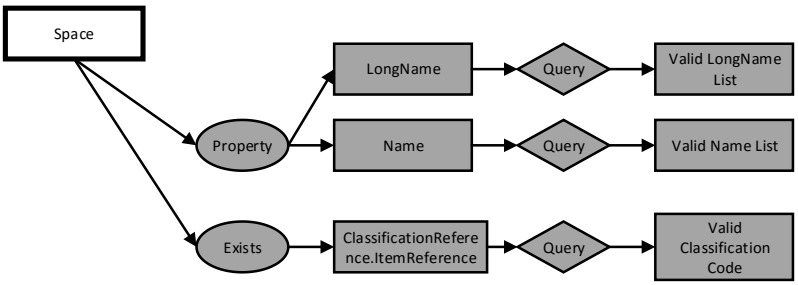


Figure 23 - Class-1 rule example

In FOL, the above rule (Figure 23) can be expressed as:

$$\forall a\big(Space(a)\big)$$
$$\wedge\, \exists a((Space(a) \wedge Query(a, Name)) \wedge \big(Space(a) \wedge Query(a, LongName)\big)$$
$$\wedge\, \exists a((Space(a) \wedge Query(a, IfcClassificationReference.ItemReference))$$

This rule checks the existence of properties Name and LongName, and checks IfcClassificationReference.ItemReference using a simple query function that checks for the existence of a property or a classification. Name and LongName properties are existing properties in IFC schema and IfcClassificationReference is the IFC entity that is used to assign classification item to an entity, which is expected in this case for an IfcSpace.

Several other examples of rules that belong to this category represented in CG can be found below:

1. "*[F] 405.10 Standpipe system. The underground building shall be equipped throughout with a standpipe system in accordance with Section 905.*" [67]

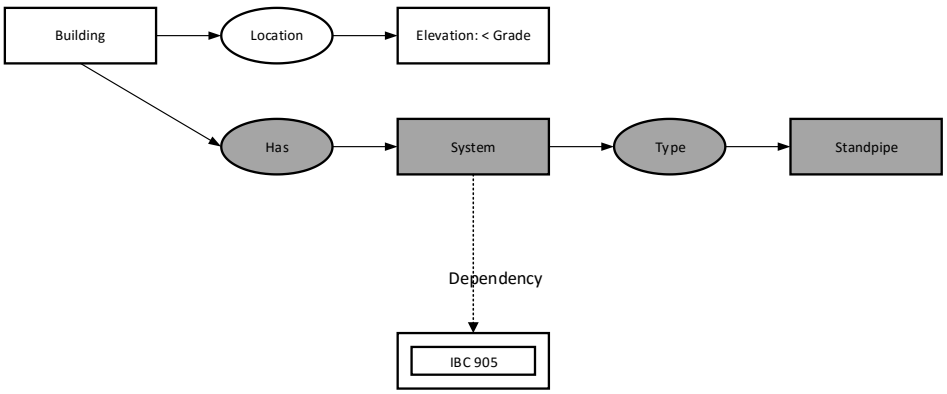


Figure 24 - CG for IBC/IFC 405.10 (ICC 2009)

2. *"Egress doors shall be of the pivoted or side-hinged swinging type"* (IBC 1008.1.2 – without exceptions) [67]

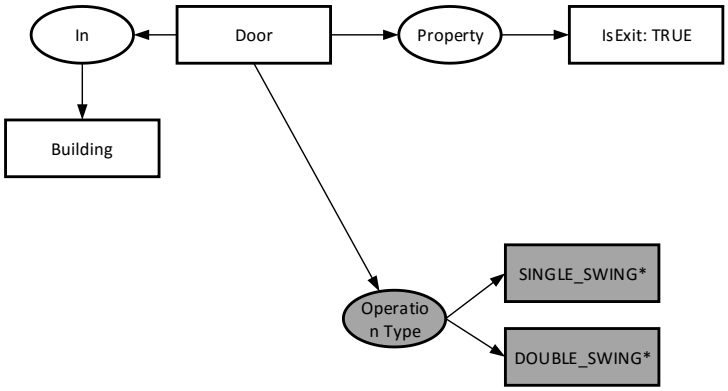


Figure 25 - CG for IBC 1008.1.2 (ICC 2009)

3. [67] "*504.2 Automatic sprinkler system increase. Where a building is equipped throughout with an approved automatic sprinkler system in accordance with Section 903.3.1.1, the value specified in Table 503 for maximum building height is increased by 20 feet (6096 mm) and the maximum number of stories is increased by one. These increases are permitted in addition to the building area increase in accordance with Sections 506.2 and 506.3. For Group R buildings equipped throughout with an approved automatic sprinkler system in accordance with Section 903.3.1.2, the value specified in Table 503 for maximum building height is increased by 20 feet (6096 mm) and the maximum number of stories is increased by one, but shall not exceed 60 feet (18 288 mm)or four stories, respectively.*

Exceptions:

1. *Buildings, or portions of buildings, classified as a Group I-2 occupancy of Type IIB, III, IV or V construction.*

2. *Buildings, or portions of buildings, classified as Group H-1, H-2, H-3 or H-5 occupancy.*
3. *Fire-resistance rating substitution in accordance with Table 601, Note d.*"

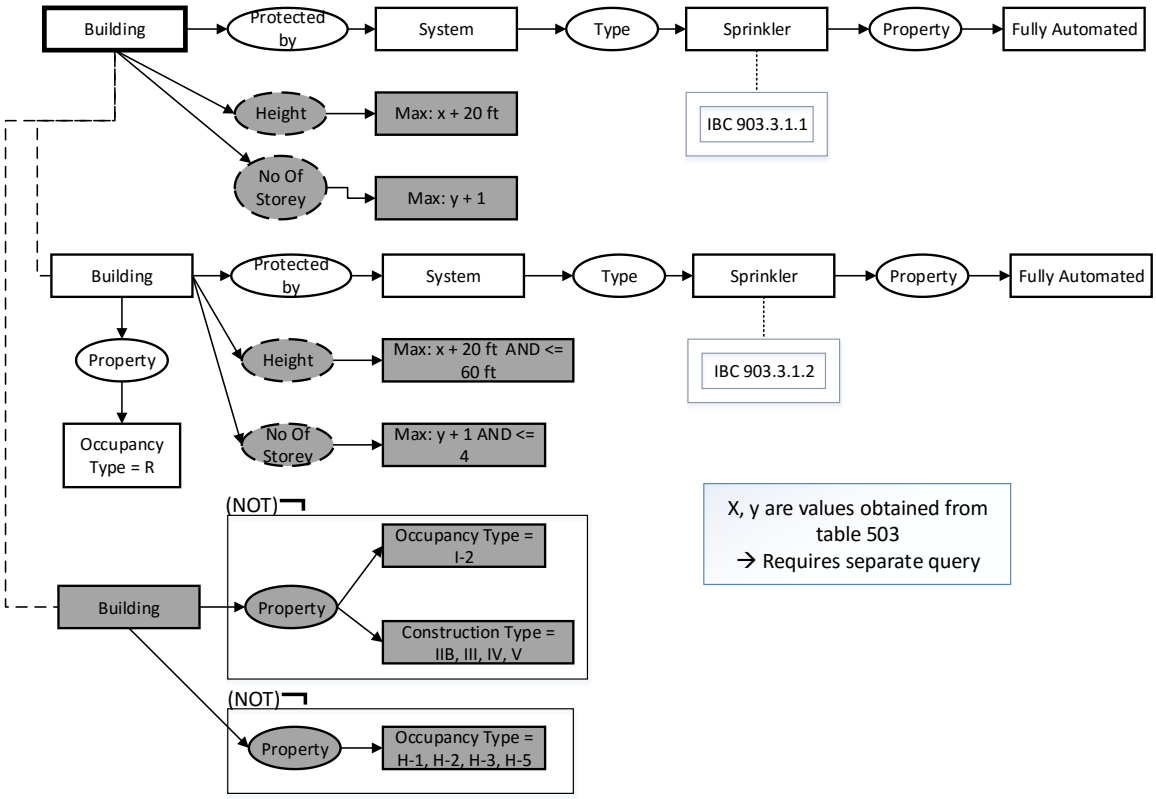


Figure 26 - CG for IBC 504.2 (ICC 2009)

4.3.1.2 Class-2 rule (Rules that require simple derived Attribute Values) example

(42) 3.2.2 Design Criteria [76]

f) The discharge pipe shall not be located in places where it can cause health and safety hazards such as locating the discharge pipe above any portable water storage tank and electrical transformer/ switchgear.

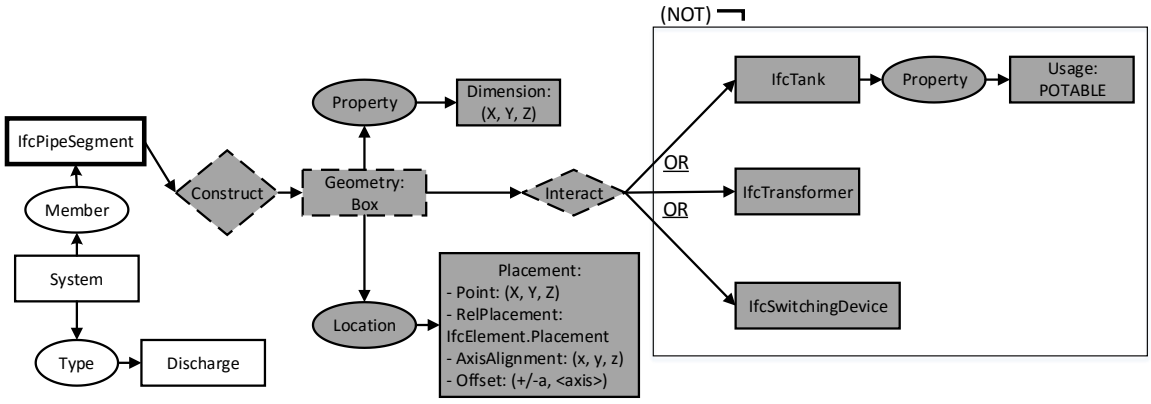


Figure 27 - Class-2 Rule Example

Representation in FOL for the above CG (Figure 27):

$$\exists b \forall a(\forall g(IfcPipeSegment(a) \wedge System(b) \wedge Discharge(g) \wedge Member(b,a) \wedge Type(b,g)) \\ \wedge (\forall c \forall p \forall d(Box(c) \wedge Placement(p) \wedge Dimension(d) \wedge Location(c,p) \wedge Property(c,p)) \\ \wedge Construct(a,c)) \wedge \neg(\exists w \exists t \exists m(Interact(c,(\forall u(IfcTank(w) \\ \wedge Usage(u,POTABLE) \wedge Property(w,u)) \vee IfcTransformer(t) \vee IfcSwitchingDevice(m))))))$$

This expression requires an extension function to construct a transient Box geometry based on a specified location and dimension. This Box is used to evaluate the existence of any type of object within the Box that is a type of a potable water tank, a transformer, or a switching device. Their existence within the Box is not allowed.
As shown in Figure 27, in this class-2 rule, we start to see the need for extensions to generate a new concept. Three such extensions are required in here: a simple box geometry, and two functions to construct the box and to perform spatial operations to find the specific object types that interact with the constructed box that is placed below the discharge pipe.

Other examples of rules in this class represented in CG are:

1. "*1106.1 Required. Where parking is provided, accessible parking spaces shall be provided in compliance with Table 1106.1, except as required by Sections 1106.2 through 1106.4. Where more than one parking facility is provided on a site, the number of parking spaces required to be accessible shall be calculated separately for each parking facility.*

Exception: This section does not apply to parking spaces used exclusively for buses, trucks, other delivery vehicles, law enforcement vehicles or vehicular impound and motor pools where lots accessed by the public are provided with an accessible passenger loading zone."

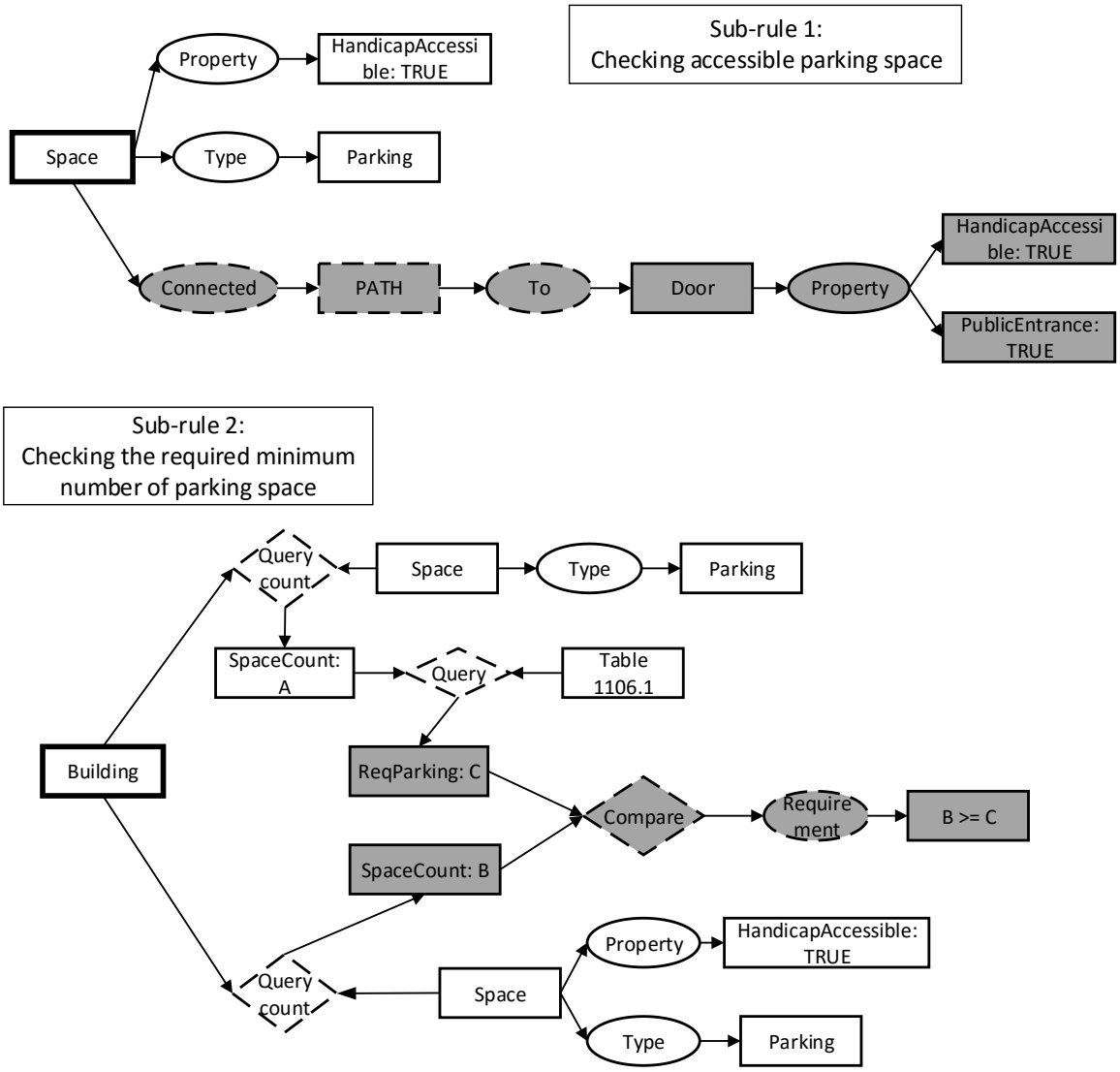


Figure 28 - CG for IBC 1106.1 (ICC 2009)

2. *"All patient rooms must be visible from the nurse station"*

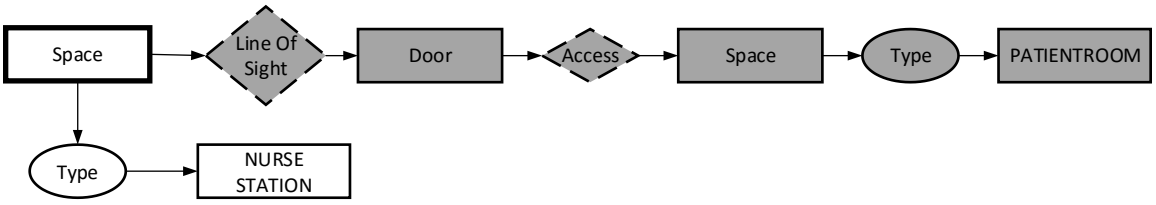


Figure 29 - CG for Patient Room Visibility Rule

4.3.1.3 Class-3 rule (Rules that require extended data structure) example

IBC 1005.2 Door encroachment. Doors, when fully opened, and handrails shall not reduce the required means of egress width by more than 7 inches (178 mm) ①. Doors in any position shall not reduce the required width by more than one-half ②. Other non-structural projections such as trim and similar decorative features shall be permitted to project into the required width a maximum of 1-1/2 inches (38 mm) on each side ③. [67]

Sub-rule #1: Reduced clearwidth by projection when door is in fully opened position

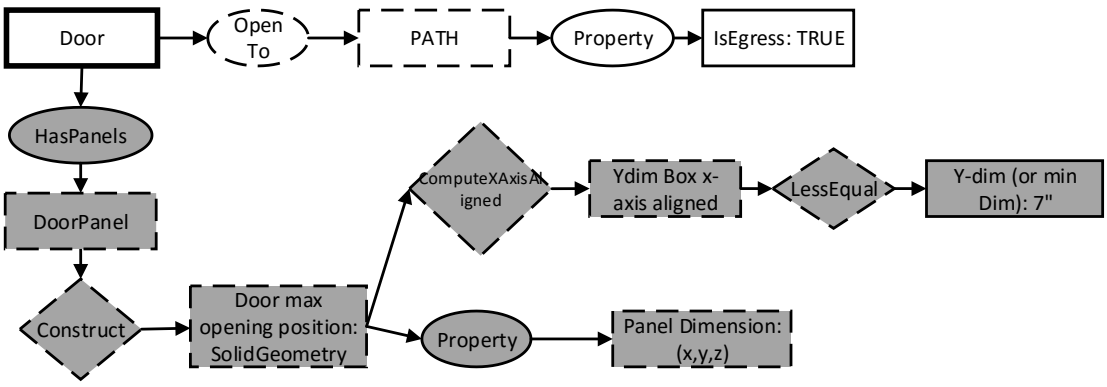


Figure 30 - Class-3 Rule Example – Sub-rule #1

In FOL the above graph for sub-rule #1 (Figure 30) can be expressed as:

$$\forall d \exists p((Door(d) \wedge OpenTo(d,p)$$
$$\wedge PATH(p) \wedge Property(p, IsEgress: TRUE)) \wedge \forall q((DoorPanel(q) \wedge HasPanels(d,q)$$
$$\wedge Dimension(r) \wedge Property(q, r: (x,y,z)))$$
$$\wedge \forall b \forall y(Box(b) \wedge YDim(y) \wedge Construct(q,b) \wedge ComputeXAxisAligned(b,y)$$
$$\wedge LessEqual(y, minYDim: 7")))$$

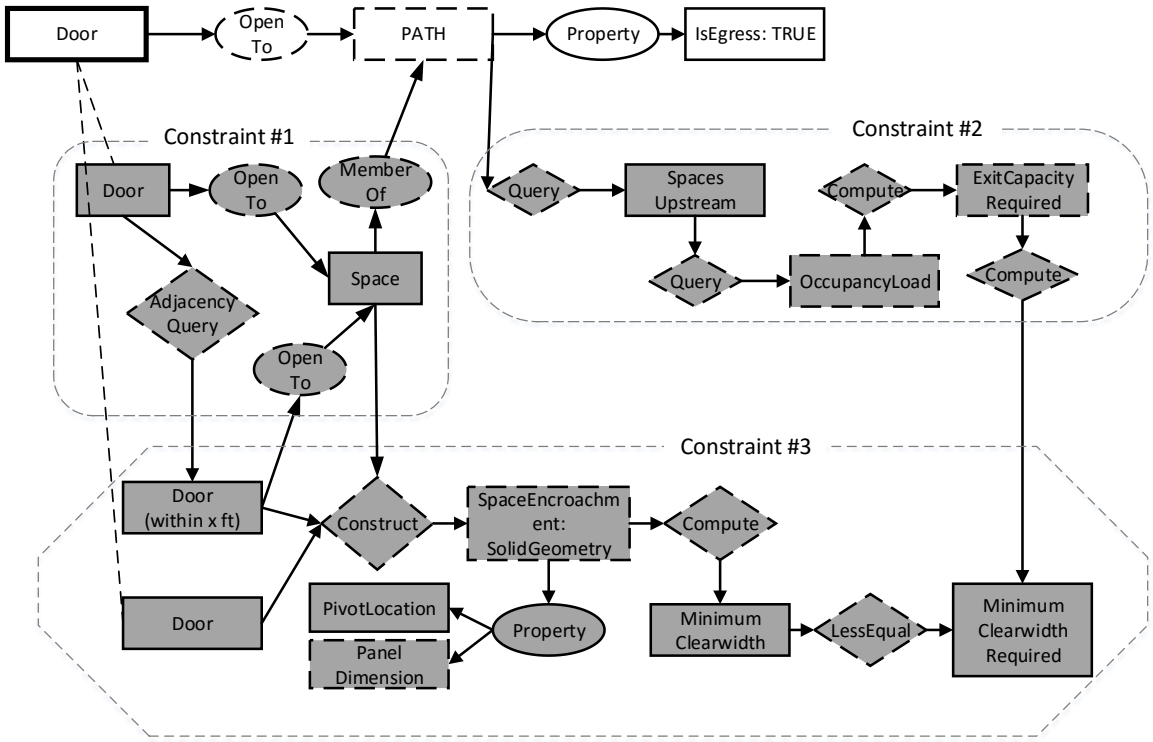


Figure 31 - Class-3 Rule Example – Sub-rule #2

Sub-rule #2 can be expressed in FOL as (Figure 31):

$$\forall d \exists p \Big(\big(Door(d) \wedge OpenTo(d,p) \wedge PATH(p) \wedge Property(p, IsEgress: TRUE) \big) \\ \wedge \exists y \exists x (Space(x) \wedge MemberOf(p,x) \wedge OpenTo(d,x) \\ \wedge DoorList(y) \wedge AdjancencyQuery(d,y,Distance) \wedge OpenTo(y,x)) \\ \wedge \forall s \forall o \forall w \big(Space(s) \wedge QueryUpstream(p,s) \\ \wedge OccupancyLoad(o) \wedge ExitCapacity(c) \wedge Query(s,o) \wedge ClearWidth(w) \\ \wedge Compute(o,c,w) \big) \\ \wedge \forall l \forall m \forall e \forall v \big(Pivot(l) \wedge Panel(m) \wedge Property((d,y),(l,m)) \\ \wedge SpaceEncroachment(e) \wedge Construct(x,(d,y),e) \wedge Property(e,(l,m)) \\ \wedge MinClearwidth(v) \wedge Compute(e,v) \wedge LessEqual(v,w)) \Big)$$

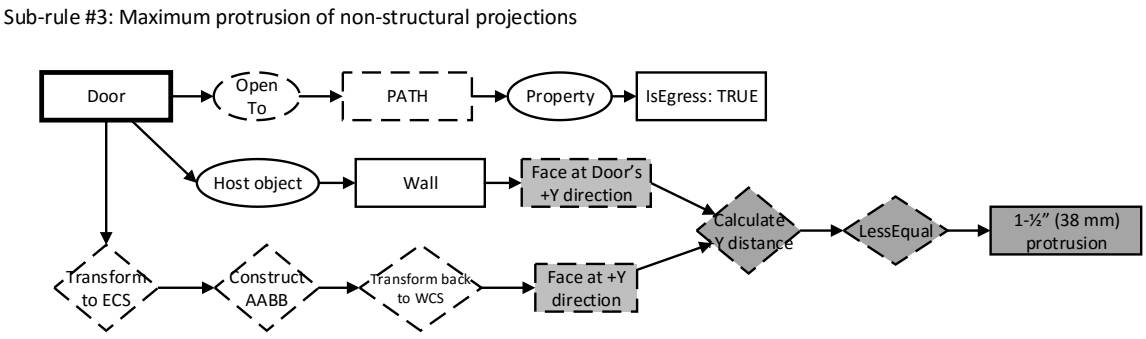


Figure 32 - Class-3 Rule Example – Sub-rule #3

Class-3 rules require extensive extensions both to the building model in form of derived data as well as functions for the purpose of computation, which includes geometry operations. The above rule highlights the nature of complexity beside the needs for extension, but also nested and branching conditions of the sentence that can occur in any entity within the statement. Sub-rule #1 sentence branches at PATH node into two constraints. Constraint #1 merges into constraint #3 at the beginning while constraint #2 merges at the end of constraint #3.

Sub-rule #2 in this example is an entirely independent rule from the sub-rule #1, except that it applies to the same main entity. Here, it is perfectly ok to separate the rule

into two rules. In other cases, the sub-rule may serve as an exception of a nested rule inside the main rule. In this case the use of the coreferent concept will become handy.

Sub-rule #3 requires a computation of the amount of space occupied by the door protrusion into the space. It can be computed using its OBB and compared to the face of the space that it is facing.

Other examples of CG for this class of rules are:

1. *"The judge's chambers are accessed from restricted circulation with convenient access to the courtrooms" [77]*

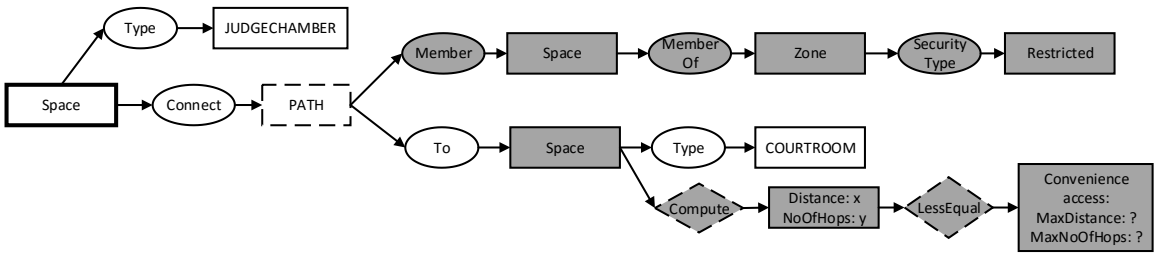


Figure 33 - CG for GSA Courthouse Rules on Accessibility of the Judge's Chambers

2. The complete IBC 1008.1.2 with all the exceptions [67]:

> ***1008.1.2 Door swing.***
> *Egress doors shall be of the pivoted or side-hinged swinging type.*
>
> ***Exceptions:***
> 1. *Private garages, office areas, factory and storage areas with an occupant load of 10 or less.*
> 2. *Group I-3 occupancies used as a place of detention.*
> 3. *Critical or intensive care patient rooms within suites of health care facilities.*
> 4. *Doors within or serving a single dwelling unit in Groups R-2 and R-3.*
> 5. *In other than Group H occupancies, revolving doors complying with Section* 1008.1.4.1.
> 6. *In other than Group H occupancies, horizontal sliding doors complying with Section* 1008.1.4.3 *are permitted in a means of egress.*
> 7. *Power-operated doors in accordance with Section* 1008.1.4.2.
> 8. *Doors serving a bathroom within an individual sleeping unit in Group R-1.*
> 9. *In other than Group H occupancies, manually operated horizontal sliding doors are permitted in a means of egress from spaces with an occupant load of 10 or less.*
>
> *Doors shall swing in the direction of egress travel where serving an occupant load of 50 or more persons or a Group H occupancy.*

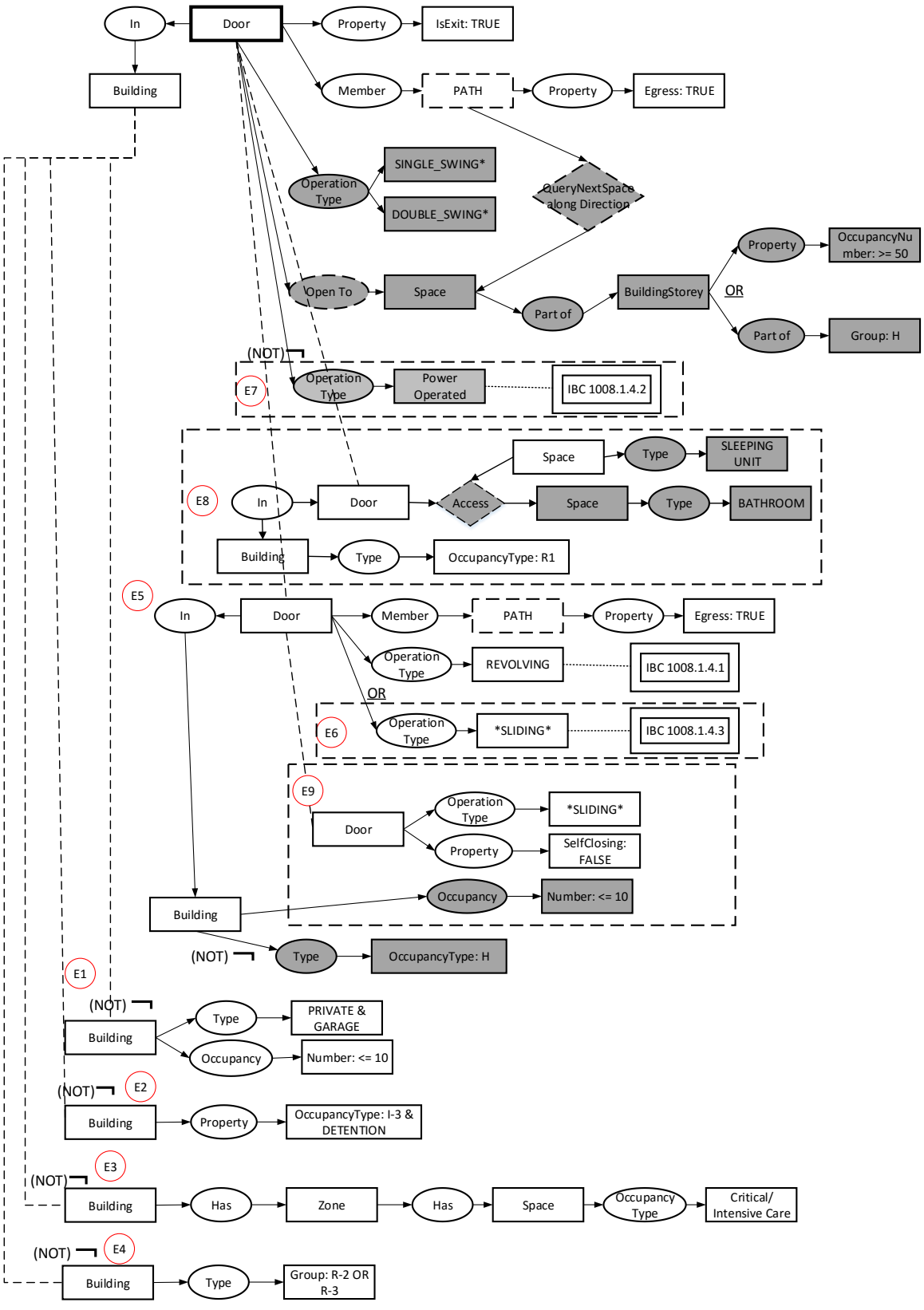


Figure 34 - CG for IBC 1008.1.2 Rule (ICC 2009)

3. 1027.1 General. (EXIT DISCHARGE)

Exits shall discharge directly to the exterior of the building. The exit discharge shall be at grade or shall provide direct access to grade. The exit discharge shall not re-enter a building. The combined use of Exceptions 1 and 2 below shall not exceed 50 percent of the number and capacity of the required exits.

Exceptions:

1. *A maximum of 50 percent of the number and capacity of the exit enclosures is permitted to egress through areas on the level of discharge provided all of the following are met:*
   - *1.1. Such exit enclosures egress to a free and unobstructed path of travel to an exterior exit door and such exit is readily visible and identifiable from the point of termination of the exit enclosure.*
   - *1.2. The entire area of the level of exit discharge is separated from areas below by construction conforming to the fire-resistance rating for the exit enclosure.*
   - *1.3. The egress path from the exit enclosure on the level of exit discharge is protected throughout by an approved automatic sprinkler system. All portions of the level of exit discharge with access to the egress path shall either be protected throughout with an automatic sprinkler system installed in accordance with Section 903.3.1.1 or 903.3.1.2, or separated from the egress path in accordance with the requirements for the enclosure of exits.*
2. *A maximum of 50 percent of the number and capacity of the exit enclosures is permitted to egress through a vestibule provided all of the following are met:*
   - *2.1. The entire area of the vestibule is separated from areas below by construction conforming to the fire-resistance rating for the exit enclosure.*
   - *2.2. The depth from the exterior of the building is not greater than 10 feet (3048 mm) and the length is not greater than 30 feet (9144 mm).*
   - *2.3. The area is separated from the remainder of the level of exit discharge by construction providing protection at least the equivalent of approved wired glass in steel frames.*
   - 2.4. *The area is used only for means of egress and exits directly to the outside*.
3. *(Omitted)*
4. *(Omitted). Exceptions 3 and 4 are omitted in this example because they are practically separate rules.*

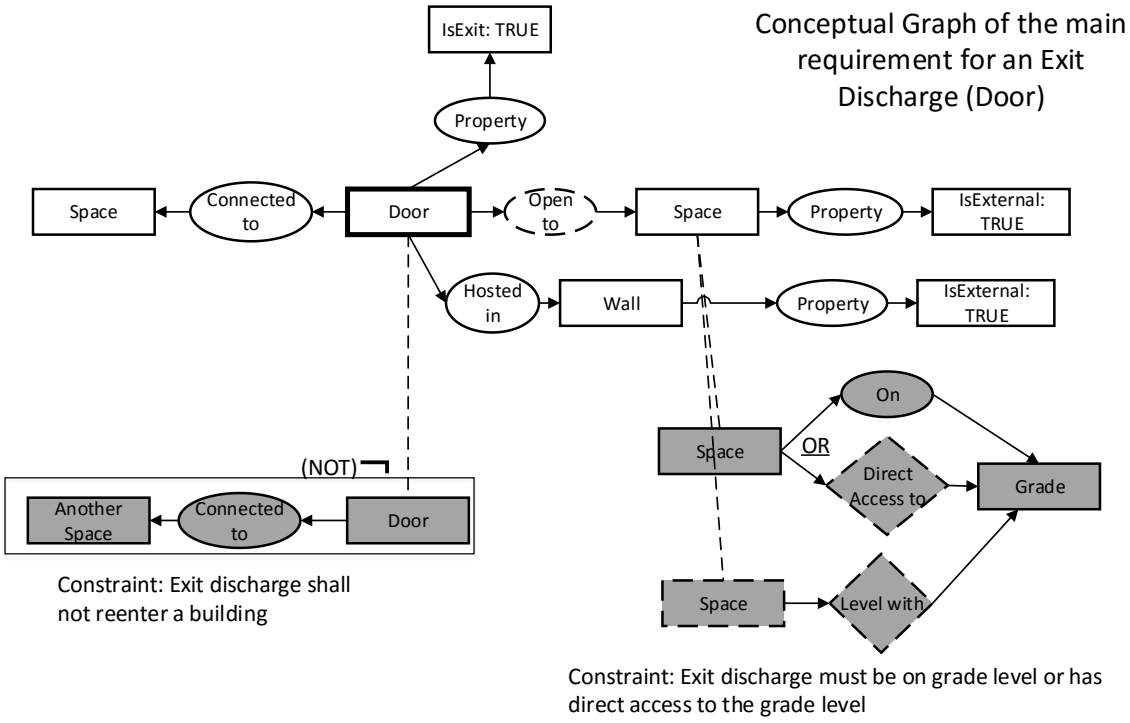


Figure 35 - CG for the IBC 1027.1 Main Rule (ICC 2009)

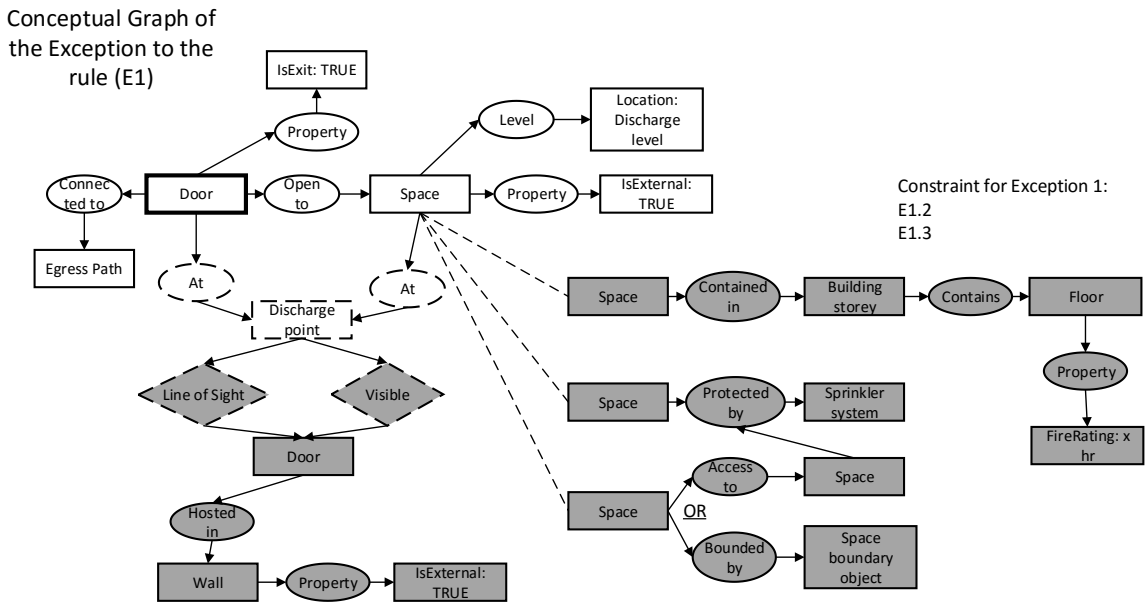


Figure 36 - CG for IBC 1027.1 Exception Rule no. 1 (ICC 2009)

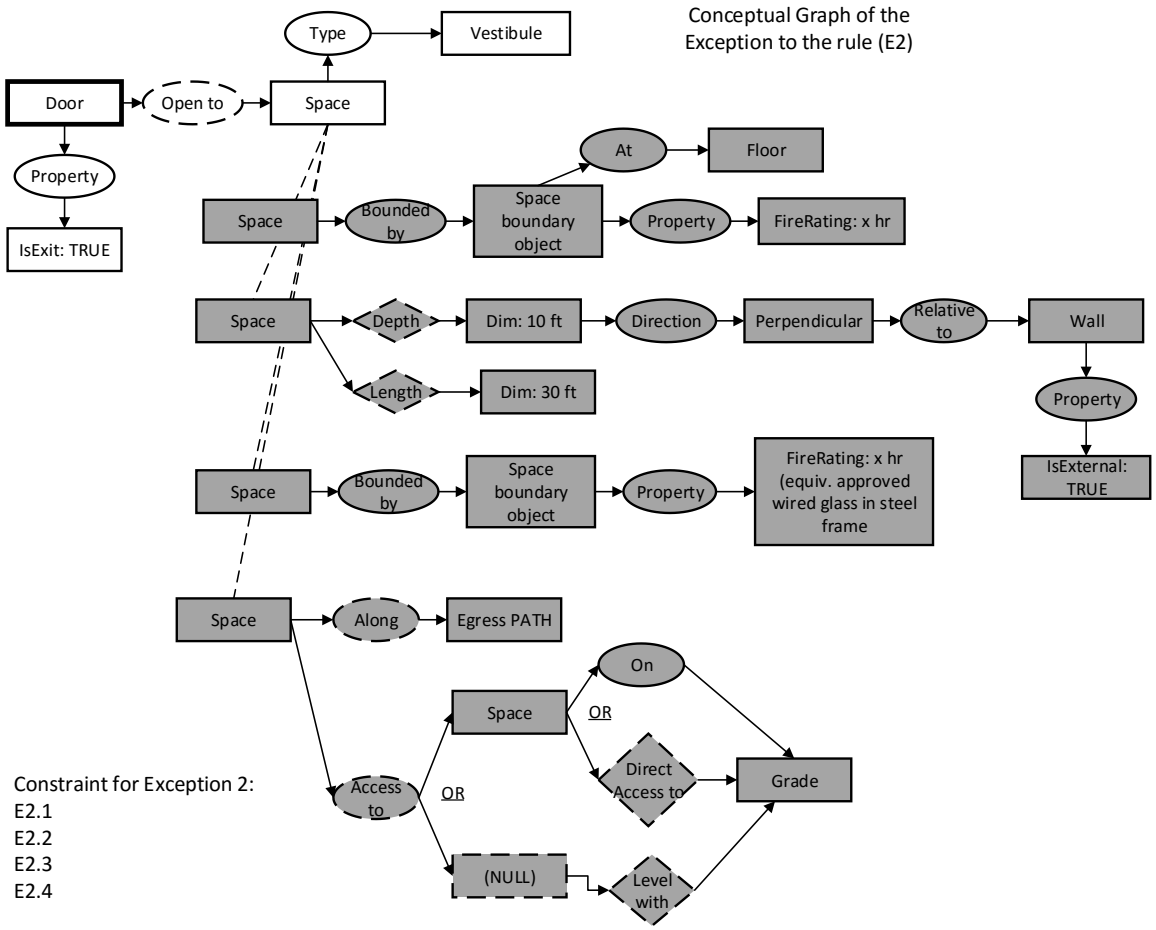


Figure 37 - CG for IBC 1027.1 Exception Rule no. 2 (ICC 2009)

4.4 Mapping the CG to the MVD

With the well-defined CG, it is possible to create a direct mapping of CG concepts and relations into an IFC MVD as well as to a UML diagram for software development (Figure 38). In the mapping to an MVD, IFC entities represented by Concept and Relation can be directly mapped to the IFC MVD, which includes relevant details such as Types and Properties. Each of the rules can be defined as one exchange requirement within the MVD. In practice, it may be practical to define only one or just a few specific MVDs, in order to consolidate the MVD requirements that often overlap among many of the rules.

Extended Concepts and Functions do not have equivalent mappings to an MVD and are only applicable for mapping to the software development environment, represented using a UML diagram in this example. Figure 38 shows an example of mapping from CG to an MVD and to a UML diagram. This example is taken from an example for a class-2 rule given earlier in section 4.3.1.2 (Figure 27).

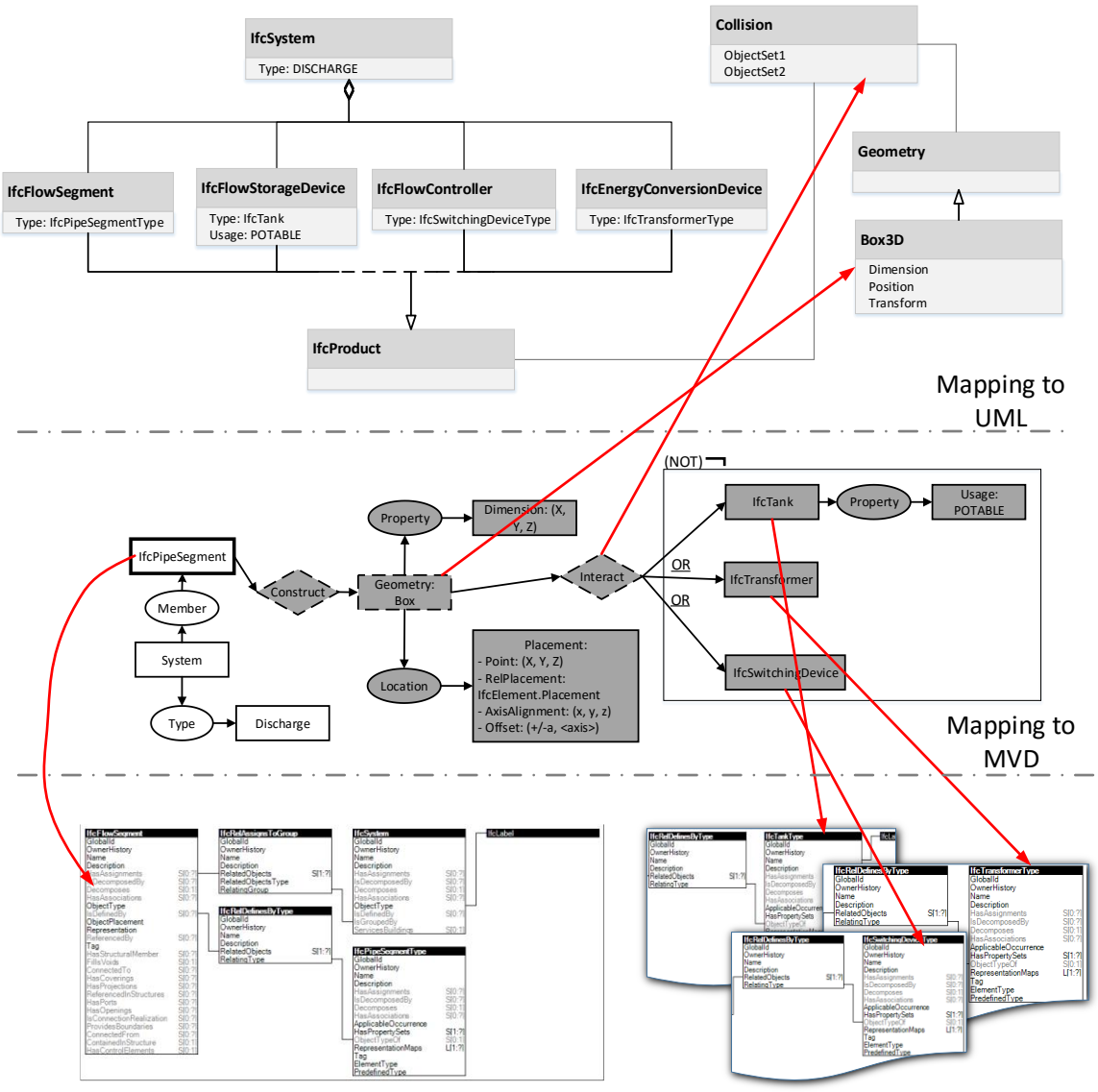


Figure 38 - Mapping CG into an MVD and UML

4.5 General Characteristics of the Rule Structure

From the exercise of describing the rules into CG and FOL, building rules range from a simple structure to an overly complex structure. The complexity ranges from man-made, i.e. the way the rule is written was probably edited in many iterations by different people over a period of time, to rules that are by nature complex due to the requirements they present. The former complex structure may be simplified by breaking the rule into smaller sub-rules. It is especially easy if the sub-rules are just a combination of various logic applicable to the same entity. An example from IBC 1008.1.2 (Figure 34) shows that several of the exceptions are applicable to the same entity for various building types or entity types. Note that the exceptions in a rule are generally rules in themselves. Below is the summary of the general logic structure of the rules:

1. The rules generally take the form of:
    a. An entity in focus, i.e. an entity where checking is to be done. For example: Space
    b. Further description of the entity, usually condition(s) that will narrow down the entity. For example: Circulation space, or Egress Path
    c. A specific entity that is defined by another entity. For example: Vestibule space where an exit discharges into (see IBC 1027.1 rule Figure 35 - Figure 37) is what Description Logic (DL) defines to be Noun Phrase.
2. Hidden in the way the rule is organized (mainly for building codes) is what building type the rule is applicable to. This is an important feature that will help users to quickly identify what rules to check for specific type of developments. Currently, this information is not captured in the examples given in this paper. They are generally independent of the logic structure and only gives further restrictions mainly for search purposes. One can imagine that this condition will simply add on to the condition at the beginning of the CG with building as an

entity and will be possibly further restricted by its building type property. In the case of mixed use buildings, a relevant constraint that is usually applied to separate building stories, can be assigned to the main object. Since CG allows the assignment of constraints to any concept node, rule definition can be tailored relatively easily to filter applicability of the rule based on the constraint defined to the specific concept node.

3. The rule requirements themselves are defined by constraints or a set of constraints applied to the entity in focus.
4. Sentence can have sub-sentences or phrases starting at any entity within the sentence. The phrase may take any of the following forms:
    a. A phrase added to the sentence
    b. A phrase added to a phrase, creating nested constraint
    c. Branch, when a phrase is added in parallel to another phrase. As part of b, a branch can have another branch creating a tree-like structure, even though usually it is not that deep. Coreference usually is represented by a branch.
    d. Merge, when two branches meet again at some point in the sentence
5. Function or Verb forms a very important part of the sentence especially at the higher level of rule classification. Functions are generally to be expected to be supported by computer algorithms in the form of a library in a rule checking system. The capability of such a rule checking system will be a direct proportion of the number of unique verbs in its vocabulary.
6. Higher level derived data is an integral part of rule structure. Many types of information required in the rules are not usually in the model and many are simply impractical to be expected to be explicitly included in the model. For example: a rule that requires a check of the distance between doors connecting to circulation space from a single room. The distance information cannot be anticipated

beforehand and it depends on the model data. Other types of information may involve information that is difficult to get manually and will be easier done by a computer. For example: a graph that represents space connectivity or access in the entire building.

7. Exceptions. Many building codes contain one or many exceptions. From a closer look an exception may take several forms:
    a. As an additional constraint to the main rule. It may be a negation or a more specific condition where the rule can be ignored or another rule is supposed to be adhered to.
    b. As a form of sub-rule
    c. As an independent rule on the same entity
    d. As a phrase that can be inserted at certain points of the sentence, often as a negation.
8. Rule dependency. Rule dependency is a rather ambiguous definition often found in the rule. It is ambiguous because the dependency is often very loose. For example if a rule has an exception that in a certain condition when there is an existence of a specific entity (e.g. a railing as form of protection), the entity must comply with rules under a certain section. Unfortunately not all requirements in those sections related to the entity are applicable for this condition and in some cases there are other exceptions that lead to a circular reference. Generally the strategy to deal with rule dependency is one of the following:
    a. Ignore it since it will be captured when the rule is handled for the referred entity.
    b. Create a dependency list for a rule checking system to automatically include dependent rules during execution. It may also need to order the execution accordingly. Care needs to be taken in this case not to create a situation where it leads to a very large dependency tree due to unrelated

dependency within the dependent rules. It also needs to ensure that it knows how to handle a circular reference.

c. Define it as a sub-rule if it is distinct enough. This may be required in a very small number of rules, especially when there is tighter dependency such as one result affecting the other.

Due to the generally complex form of rules, especially building codes, it is generally a good idea to try to separate one large rule into independent sub-rules. It will make the rule more discernable and hence easier to maintain. This is beneficial even if there are small redundancies between sub-rules due to some minor variations in the sub-rule. Many exceptions as described above can be either separated as an independent rule, as a sub-rule or inserted into the main rule.

4.6 Suitability of the Typical Rule-based System for BIM Rules

Having identified the characteristics of building rules, the next question that may be asked is: what is the difference between these rules with the typical rule-based system or general knowledge based system? This question is important because the answer will determine the future direction of automated rule checking definition. The same question seems to be in the mind of many other researchers working on the issue. For example, Beach proposed the use of the open source rule engine DROOLS as the execution environment of automated rule checking [22], and Zhong used the JESS rule engine [62]. It is important to examine whether the rules in both definitions are actually the same. Both DROOLS and JESS are derivatives of RETE match algorithm introduced by Forgy [93, 94]. The Rete match algorithm is designed to work with:

1. Efficient search of a fact against a large number of rules, which themselves are in the form of facts and conclusion. The general form of the rule is: ***IF*** *<condition or fact>* ***THEN*** *<conclusion>*

2. The search is not linear, but it is nested. The search basically traverses the built in knowledge base (the rules) like it traverses a tree with many branching conditions. It needs to deal with potential conflicting conclusions from the different branches being searched for the given fact. This is part of the reasoning process. For example, a person with a university degree qualification, married with 3 children, earning $100,000 annually, with an existing mortgage commitment of $250,000 for the next 15 years: is this person qualified for a new property loan and if he is how much is the maximum loan amount that he is entitled to? This example requires rather complex searches that will have to go through many defined rules, which may be nested due to the different risk profile.
3. The facts in the rules are all well-defined.
4. Such a knowledge base also deals with inference, e.g. given the conclusion, state what the conditions are that must apply. For example: if a person applies for loan of $100,000, what would be the requirements that the person must fulfill to get the loan application approved?

On the other hand, building rules have the following characteristics:

1. A large number of rules are applicable to specific entity types with some conditions.
2. There may be two levels of applicability, first on the building type and second on the specific entity
3. With the exception of rules in class-1, the rules are expected to be in form of a "template" rather than in its final form. The template is essentially the same rule definition but with some values defined using variables that can be specified at runtime, or even assigned to specific values that can be queried from the model. Using a template approach, a rule can be made more generic. This is very useful for some rules that perform the same basic logic but for different objects or building types.

4. The rules can be broken into self-contained atomic rules that can be run independently. A complex rule that consist of multiple atomic rules, which may influence how the outcome of the other atomic rules are to be evaluated, can itself be a rule that make use of the outcomes of the atomic rules and apply specific logic to determine the final outcome for the rule. Such nested dependency requires outcome from one rule to be kept temporarily until all the other dependent rules are completed for the main rule to use them. In some cases, an outcome of one rule is needed for other independent but related rules. In this case a temporary storage of such outcomes may be needed with a lifetime beyond the runtime of the rule itself.
5. There is hardly any inference reasoning needed, except for a specific requirement to search applicable rules for a specific building type.
6. In many of building rules, variables in the templates are filled with values that are obtained from the building model that is being checked. This circular dependency, where the building model is subjected to the rules and the rules derive some values from the model, is peculiar to building rules. This makes the rules to be considered as not well defined until runtime.

Different characteristics of the traditional rule-based system and BIM rule checking system described above is illustrated in the Figure 39 below. In the BIM rule checking system, the search is performed on the data to filter suitable objects that are applicable to the rule. This requires a query system to the data. The second important distinction is the concept of the derived objects or attributes. This has been highlighted in CHAPTER 3 and it is represented as derived concept nodes in the CG. From these pieces of information, the clear answer to the question poised at the beginning of this section is that the traditional rule based system can be used as a platform to implement a building

rule checking system, however it does not really address the real issues that are to be solved with building rules.

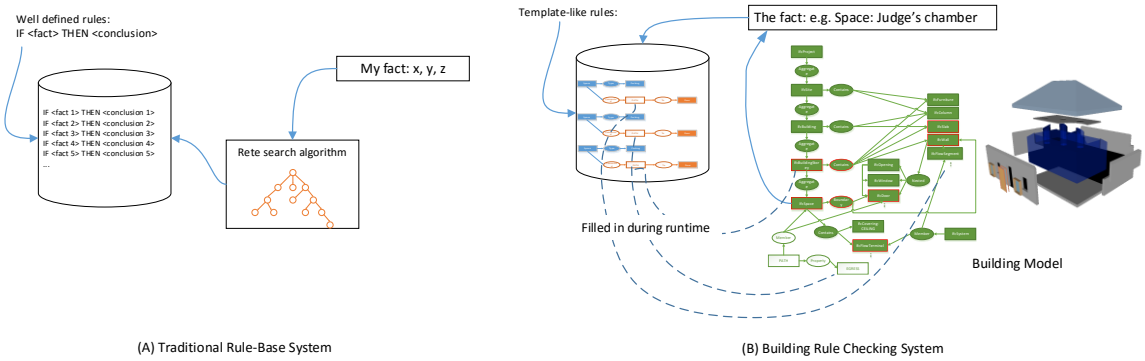


Figure 39 - Comparison of the Traditional Rule-Base System and Building Rule Checking System

From the analysis of rule complexity and its structure, the suitable approach seems to be providing the following key concepts:

1. Providing an efficient query system that allows a flexible and efficient query to be performed on the BIM data. It is both for the purpose of direct checking for rule requirements, especially in the case of rules that belong to class 1, and for the purpose of filtering suitable objects for further checks. The queries are hard to pre-define without making the rules extremely rigid and tailor made for only specific types of checking. CHAPTER 5 describes an efficient query-able database that is possible for BIM data and is suitable for rule checking systems.
2. Support for geometry data and spatial operations. This is a challenging requirement since traditionally it is not easy to provide such support. The encouraging fact about the geometry and its spatial operators is that the supporting theories are well defined and there is a successful precedent in the 2D applications in the GIS domain. CHAPTER 6 describes the unique concept of integrating the support for 3D geometry in the database, its indexing scheme and the multiple representations required to support a BIM rule checking system.

3. The third important concept is the definition of a standard rule language specifically designed for BIMR rule checking in mind. It integrates the query-able BIM database, geometry and spatial operators, the concept of transient geometry, a potential extension for a graph in the database, and shortcuts to certain well defined relationships defined in IFC schema. CHAPTER 7 describes the details for BIM Rule Language, its design components, its syntax and its use in a BIM rule checking system.

CHAPTER 5

QUERY-ABLE BIM AND ITS SCHEMA

BIM data is essentially a database that contains all the alphanumeric-based data about buildings plus their geometries, and spatial and topological relationships. At present, to achieve runtime efficiency, all the BIM data are managed and contained inside the proprietary formats. As a result, BIM data is locked inside the vendor specific implementation and interfaces [95]. The only way to access the data is through standard interfaces that each of the software developers provide in their respective tools. In many cases for a need beyond the standard interfaces, additional access is provided through scripting or through a set of APIs. The problem is that these methods are proprietary in nature, which means that an application written for one product will not be applicable to another. Also the effort required to access the data vary from relatively straightforward methods that are however limited by predefined interfaces and tools, to complex methods via APIs. Since one of the most valuable assets in any database is the content itself, the hurdle to mine the BIM data is a serious problem that needs to be overcome.

The problem of access to the data is not unrecognized. Attempts have been made to enable access to the data more easily. Each vendor as mentioned above generally provides the standard or pre-defined functionalities to get into the data, for example using schedules. This approach is limited to only the objects with their properties. For more sophisticated data, one needs to write custom programs using the APIs provided. All these limit the kind of data that are available. Ad-hoc queries are severely restricted, which does not allow discovery or insight from the data besides those already pre-determined. One promising approach to deal with this issue is to use a kind of database that allows access to the data outside of the vendor specific APIs. Most work in this area involves the standard exchange model based on STEP and IFC (which is derived from

STEP). This effort has been explored as early as the1990s. In his PhD thesis, Loffredo assessed the implementation of standard data access interface (sdai) performances of various Express to database mapping. The mapping itself had been explored by various authors since the late 1980s [95]. Since the IFC standard was released in 1996, there have been several developments towards a model database or IFC model server. The BLIS project was among the first ones which aimed to develop the concept of standardized access to BIM data in the server via Web-Services. It was based on the relatively short-lived IFC 2.0 specifications [96]. In 2002, Adachi developed an IFC Model Server that was developed using an SQL Server database and provided access to the data via Basic API and ADO interface as well as SOAP web services [97]. In 2004, Eastman et al developed GTCIS2SQL to translate STEP based data into an SQL database [98]. While this focused on the CIS/2 schema, the transformation worked for general STEP based data, including IFC. In South Korea, a project developing an IFC model server took a slightly different approach with an Object-Relational approach using the commercial Object-Relational Management System CUBRID [99]. The more recent development of an IFC based model server was done in the Netherlands producing an open source BIMserver that is based on a column oriented BerkeleyDB [41]. It comes with the accompanying idea of a standardized query language similar to SQL called BimQL [42]. In the commercial front, Jotne EPM is probably the most active vendor pushing for an IFC Model server that is based on their established Express Data Manager that is used widely in the PLM domain [100]. It is based on their proprietary object based database.

As noticed all the development towards IFC model servers are based on databases, with the relational database being one of the dominant options. However, performance issues have been documented related to the use of a relational database for IFC model data, especially when it involves complex queries. Lee discussed the issues in [101]. He also presented an alternative implementation making use of the relatively more recent trend of the use of Object Relational databases. Such a system appears to have

advantages over a purely relational approach as reported in a performance comparison using the standard BUCKY performance benchmark [102]. Nevertheless, in general the performance is still poor compared to a standard RDBMS due the complexity of the IFC model.

Most of the development of model servers has focused on storing the STEP data into some form of database, preserving the original STEP data structure. IFC, which is based on STEP, has the same issues. The differences between the hierarchical nature of object oriented data and the relational data is probably the single most important cause of the performance issues. The current approaches have to deal with a relatively large number of entities in the relational database. In IFC2x3, there are 653 entities and 327 types (117 defined types, 164 enumeration types and 46 select types). In IFC4, the numbers have grown to 766 entities and 391 types. Thus, creating 1-to-1 mapping to a relational database clearly imposes a heavy burden on the relational model. It also requires a large number of tables in a join statement just to be able to access even relatively simple data. The select types also cause additional complexity for queries as they have to be evaluated first before further queries can be made to the data.

In this thesis, a different approach to the BIM data is taken. Instead of taking the viewpoint from database management aspects of the complete IFC entities, thus preserving all the entity definitions, or from the perspective of data exchange where entities must be explicit to ensure a well-defined and precise exchange, a point of view from the perspective of the users of the data is considered. For the most part, users will be interested in finding out about building objects, their properties, types, geometries, spatial, and other relationships. For example, they do not need to know how to re-construct the geometry, but they need the final geometry. In fact the most complex part of the IFC schema is generally related to the geometry definitions. The users of the data generally want to know the information about the model, how to find objects of interest that are often spatially related such as where a certain object is located, and how objects

interact with each other. All these should be done in a simple and a standardized way, reducing reliance on scarcely available technical expertise of the data model, and instead focusing on the data users are familiar with.

In the general field of database management systems, a similar concept has been developed for many years, known as a data warehouse concept [103, 104]. This similar approach to BIM data is used and BIM data is transformed into a database schema that is intuitive and efficient to be queried. In the rest of the chapter, the content is organized in the following manner:

- General overview and approaches to the data warehouse concept
- The detailed description of the data warehouse-like schema for BIM data
- Issues of geometry and spatial support for BIM database
- Review of database management systems supporting 3D geometry and spatial operations
- A demonstration of how the database supports queries for rule checking requirements is described in CHAPTER 4. This is done using examples to validate the schema for its suitability and performance running such queries.

5.1 General overview of the data warehouse concept

The main motivation for a data warehouse concept is a recognition that information is the most important asset to any organization. It concerns how to get the data out effectively and efficiently in a form that is logical to the business or end users. Kimball [104] and Adamson [103] cover at length the introduction to a data warehouse concept in the earlier chapters of their books in an easy to read introduction into the topic. In essence the goals of a data warehouse involve the following key items as defined by Kimball [104]:

- The data warehouse must make an organization’s information easily accessible
- The data warehouse must present the organization’s information consistently

- The data warehouse must be adaptive and resilient to change
- The data warehouse must be a secure bastion that protects the information assets
- The data warehouse must serve as the foundation for improved decision making

The data warehouse is linked directly to an analytical system that provides access to the data in easy, consistent and flexible manner, which is known as OLAP (online analytical processing). In short, a data warehouse is a database devoted to analytical processing [105]. The drive towards the above stated goals leads to a simple style schema known as a star schema. A star schema is characterized by a focus on a central fact table and a set of surrounding dimensional tables. One example of the fact table (Order Facts) and its dimensions from chapter 1 of [103] is shown in Figure 40. By a strict definition, the star schema is an acyclic database schema having a join tree of a depth one, i.e. there is only one step away to get into all the dimensional information from the fact table. The star schema differs greatly from the operational data, which is typically designed using Entity Relation (ER) with the third normal form (3NF) [103]. Transforming an operational data into a data warehouse star schema requires definitions of the key components of data warehouse information (or data mart) that will form the fact table and its dimensions. While generally there is no single method that can be used for all business requirements, the data warehouse schema often requires the following approaches:

1. The fact table represents a business process that is a natural business activity performed in an organization. In a typical data warehouse system, the fact table is where all the measurement is captured. Since the focus is on a particular business process, the fact table is usually an entirely different table than those in the production database. This typically requires a pre-processing step to collect the relevant facts from various sources of the production database. The process is known as an ETL (Extract, Transform and Load) [103]. This process will be described in more detail later.

2. Dimensions provide context to each of the measurements in the fact table. They provide details on a specific topic. Because the dimensions could be complex, in a star schema it is often required that the various information contributing to the dimensions are de-normalized into a flat or single table.

The star schema looks deceptively simple, but it is widely used successfully in practice [104]. The advantage of the star schema is its simplicity that reflects the business focus. The depth of the join relationship makes it easy also for a very efficient query. However, in many cases a simple star schema often is not sufficient to capture all the dimensions, or it may create too much redundant information in its de-normalization process. It leads to a variation of the star schema to a normal form for the dimensional tables known as a snowflake schema (Figure 41). The snowflake schema provides more intuitive meaning to the dimension tables and avoids redundancies [106], but it may also sacrifice the simplicity and efficiency of access to the data [105].

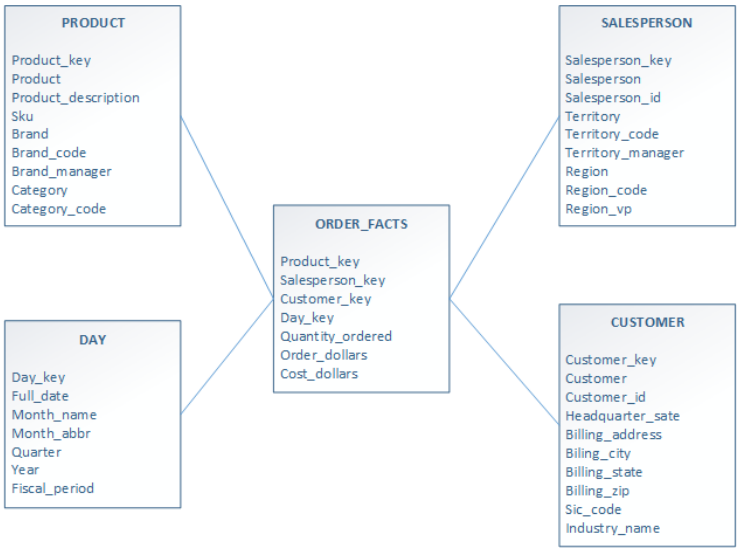


Figure 40 - A star schema [103]

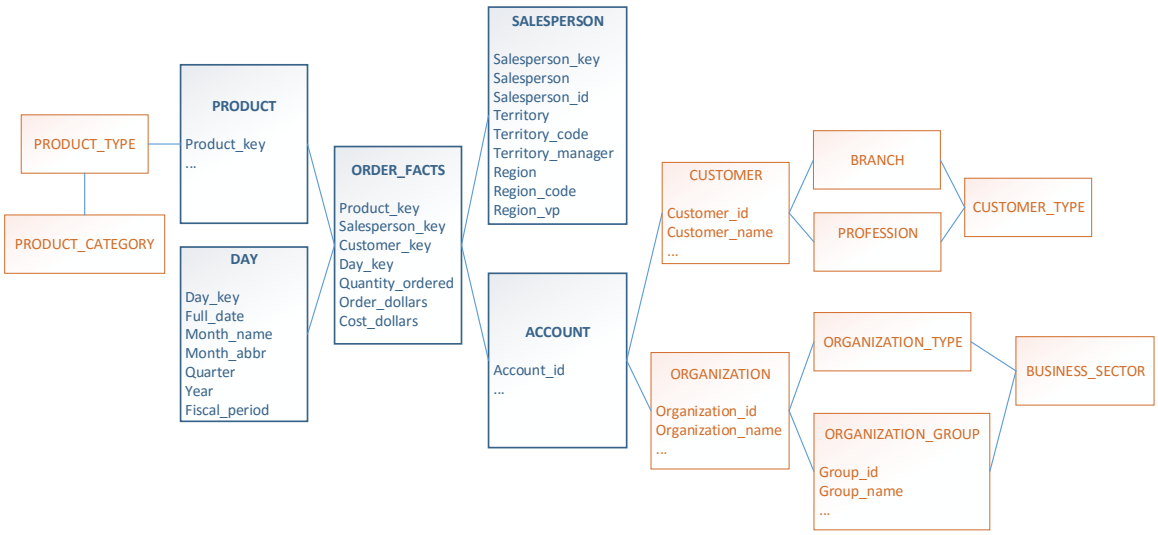


Figure 41 - A snowflake schema (adapted from [103])

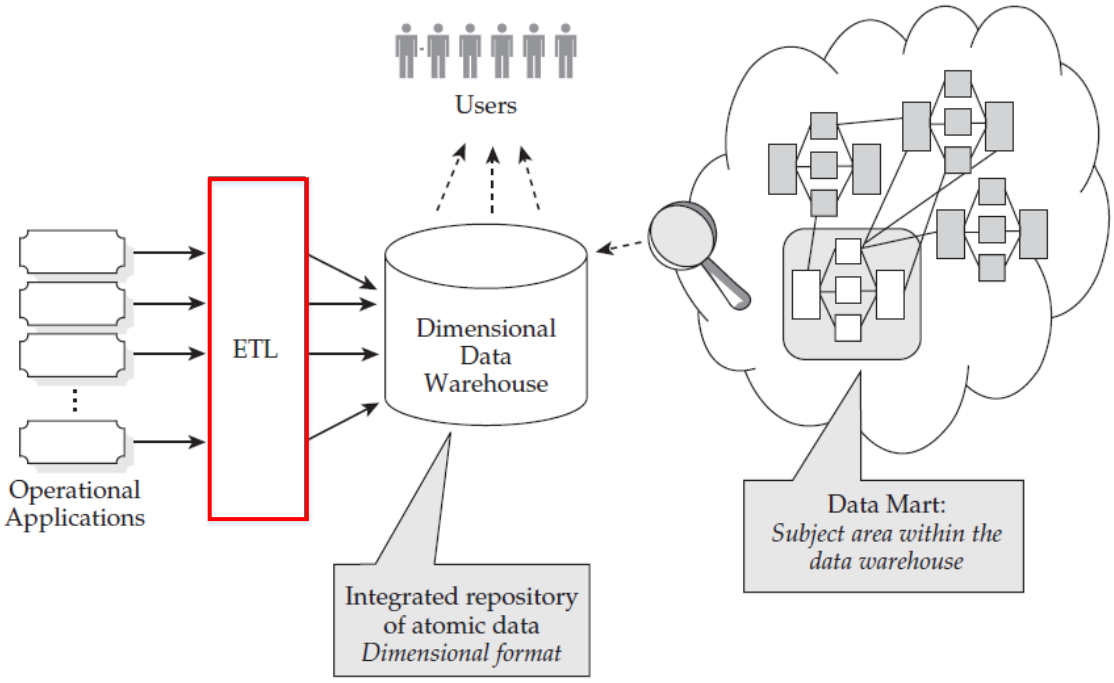


Figure 42 - A data warehouse architecture according to Kimball [103]

Data warehouses contain information that focuses on getting out the data (queries) and they are mainly static data or read-only in nature. To feed the data into the system, the process required is known as an ETL (Extract, Transform, Load) (Figure 42). ETL

essentially defines mapping rules from various sources into the data warehouse and it is a nontrivial process that is usually done in batches [103].

5.2 A data warehouse like schema for the BIM data

In many ways BIM data possesses many characteristics of a data warehouse. It certainly shares the concerns about easy access to the data, simplicity of the data for efficient access, and its focus on the end users' logical view of the data. In many applications, access to the basic building model data can be restricted to read only, e.g. for rule checking, analysis, and even for facility management purposes. Instead of focusing on the IFC data structure, we want to focus on what really matters from the user's perspective. This leads to a data warehouse-like schema for the BIM data with the characteristics described below. It is called data warehouse-like because it is not exactly identical to the star nor snowflake schema definitions in the data warehouse domain, but it has a strong resemblance and many of the principles in the data warehouse domain are applicable. The database schema is named BIMRL and reflects the research on rule checking systems for BIM rules.

- There is a need for easy and efficient access to the BIM data. This is expected to be performed by the building professionals themselves rather than the technical specialists. This is increasingly important because of the large amount of data being captured in BIM.
- The need for analytical capability in the BIM data that cannot be restricted to a set of narrowly pre-defined set of functionalities. What is needed is the ability to support an open-ended set of ad-hoc queries that may change at any time depending on the business need. Such access should be available immediately as soon as the BIM data is available. One usually cannot wait for weeks or months for the development of a new feature to gain access to the information.

- A central fact table can be identified for the BIM data. It is simply called a BIM element or BIM object (BIMRL_ELEMENT). Generally it captures the building object as part of IfcProduct (IfcElement, IfcSpatialElement), but it also includes IfcGroup (IfcSystem and IfcZone). All the common attributes shared by most of these objects, if not all, are aggregated into the fact table. Only the specific attributes that belong to the individual entities are kept in a separate property table grouped under a virtual property group named IFCATTRIBUTES. This table is shared with the property set table. Figure 43 shows the details of the BIMRL star/snowflake like schema for this purpose. There are a few deviations to the typical data warehouse schema. They are described in more detail below. (The higher resolution version of the schema diagram is attached in APPENDIX A).

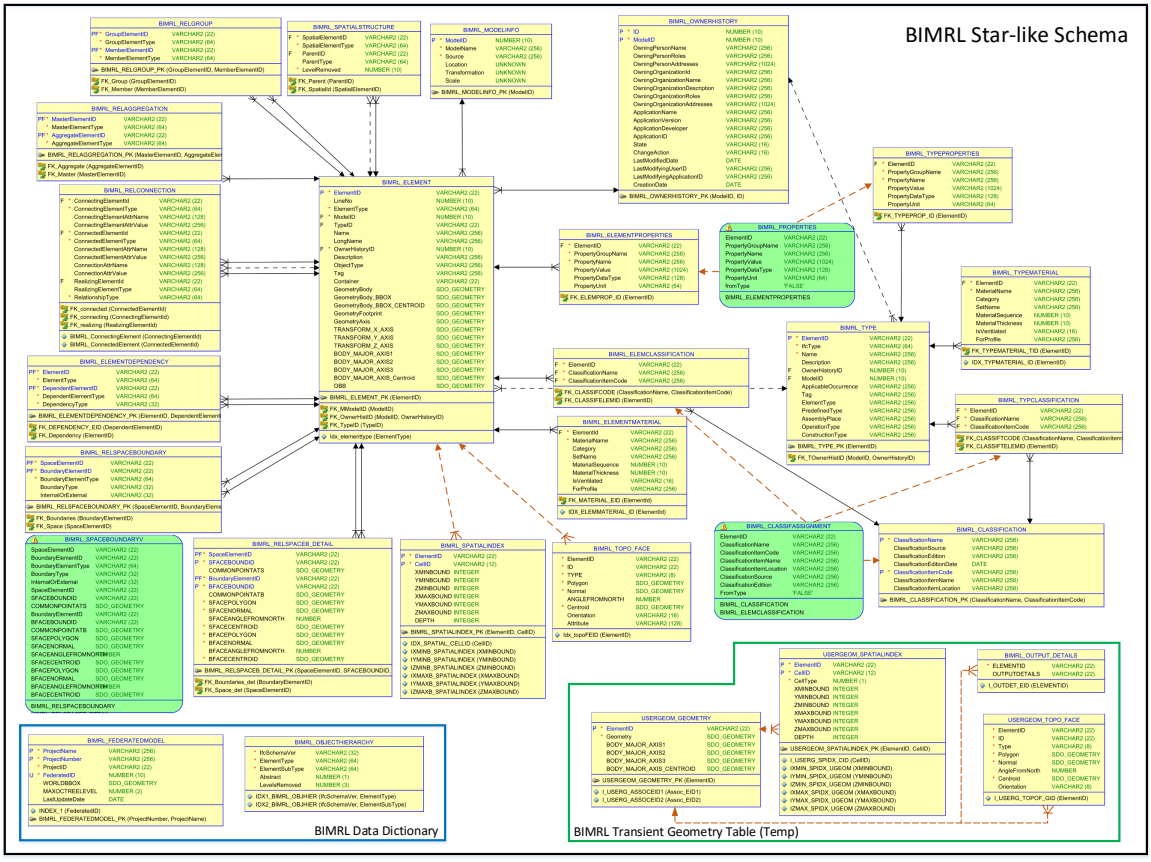


Figure 43 - BIMRL Schema

- Distributed model concept.

  Dealing with a large number of records in the database requires very good tuning. However, with the increasing number of records, performance will be affected. In the traditional model, the BIM models may be inserted into a single set of tables. In BIMRL, each set of federated BIM models will have its own set of tables, independent from other models. In this concept, performance for each model remains constant and only depends on the number of records within the model and not on other models in the database. The tables are named with the federated model id (in hexadecimal format) as a suffix. For example, the main fact table BIMRL_ELEMENT has the actual name BIMRL_ELEMENT_002E for a model with federated model id = 46 (or 0x002E).

5.2.1 Scope of the research

The scope of this research currently focuses on the building elements and all their associated information such as properties, types, geometries, and relationships. In terms of IFC coverage, this research uses and conforms to the scope of IFC2x3 Coordination View 2.0 MVD [65]. This MVD is currently the most general purpose MVD defined and it is the basis of most BIM authoring tools' support for IFC export and import. The choice is deliberate since the building elements are the main artefacts that will continue to live throughout the lifecycle of the building and most of the time they are the focus of query and checking requirements. They also are the more readily available information today, relatively mature, and supported well by most BIM authoring tools. As the consequence of the conformance to CV2.0 MVD, process related entities are excluded in this research scope. A short discussion on the possibility of extending this methodology to the process related entities in future works is described in a later section of this chapter.

The choice of CV2.0 MVD is not a limitation of this approach, it is just a means to limit the scope of work and to allow the work to be based on a well-defined, generic, and widely supported specification by the major BIM authoring tool. It does not automatically exclude other MVDs since the concept is rather generic, rather, it is simply not verified to work within the scope of other MVDs. Extending support for other MVDs may involve modification to the implementation but should retain the same concept.

5.2.2 Detailed information of the fact table – BIMRL_ELEMENT

The main fact table is named BIMRL_ELEMENT and keeps all the IfcProduct entities, including IfcGroup as mentioned above. In addition, there are two tables that serve as detailed tables to the fact table and are kept as separate tables due to the variable number of records that makes it impractical to capture them inside a single fact table. They are:

- Properties grouped in property sets (BIMRL_ELEMENTPROPERTIES)
- Materials (BIMRL_ELEMENTMATERIAL). Ideally materials should be kept in a separate dimension table. However, in the current implementation, the use of materials is still inconsistent with many redundancies. Since the main purpose of this approach is for read-only query focusing on the fact table, we could live with the redundancies by keeping materials as individual properties of the objects.

To ensure data integrity and consistency, the de-normalization process within ETL must maintain a consistent view of the data by observing the following rules on properties. The same principles apply to other types of data.

In the given sets of objects (O) and property sets ($\wp$), there are distinct set of objects $O = \{O_1, O_2, O_3, \dots\}$ and $\wp = \{\wp_1, \wp_2, \wp_3, \dots\}$, where the property set is in turn a set of properties $\wp_i = \{P_1, P_2, P_3, \dots\}$, that follows:

1. There is no redundant Property within a specific property set: $\forall P_i \in \wp_x, where\ P_i \neq P_j\ for\ all\ j \neq i$. This also implies that property P_i can be a member of multiple property sets: $P_i \in \wp_1, \wp_2. \wp_3$, ..., though such usage is not encouraged due to potential confusion.
2. There is a master property set definition that contains the aggregate of all the properties within the property sets:

   $\wp' = \{P_1, P_2, P_3, P_4, \ldots\}$, where $\wp_1, \wp_2, .. \wp_n \subseteq \wp'$
3. Each object O can have one or more property sets attached to it:

   $$\mathcal{P}(O) = \{\wp_x, \wp_y, \wp_z, \ldots\}$$
4. In the mapped system, property set attachment must be conserved to maintain consistency of the property set assignment but rule no. 3 will not be fully maintained. The physical relationship will not be maintained, only the logical one is. For two objects O_i, O_j that are assigned to the same sets of property sets:

   $\wp(O_i) = \{\wp_{1i}, \wp_{2i}, \wp_{3i}, \ldots\}$ and $\wp(O_j) = \{\wp_{1j}, \wp_{2j}, \wp_{3j}, \ldots\}$, where $\wp_{1i} \subseteq \wp'^c, \wp_{2i} \subseteq \wp'^c, \wp_{3i} \subseteq \wp'^c, \wp_{1j} \subseteq \wp''^c, \wp_{2j} \subseteq \wp''^c, \wp_{3j} \subseteq \wp''^c$, where $\wp' \equiv \wp'^c \equiv \wp''^c$ and are redundant copies.

5.2.3 Dimension tables

5.2.3.1 Relationships data

The IFC schema defines various relationships. There are 49 entities in IFC2x3 (47 in IFC4) that are derived from IfcRelationship. One of the major aims of a data warehouse is to simplify the data model (see Figure 43) and denormalization is always required to achieve it [107]. In dealing with IfcRelationship entities, semantics of relationships is used as described in Table 5. This is used to guide the selection of five relationship tables and 2 other forms of relationship data. It is important to note that

unlike typical data warehouse data, BIM data has the unique characteristic of relationships between the objects that points to itself (Figure 44). This relationship can be described as: $r \rightarrow f_{(x,y)} = \{x, y | x \in \mathbb{E}, y \in \mathbb{E}\}, where\ x \neq y\ and\ \mathbb{E}$ is the set of building elements represented by BIMRL_ELEMENT table. Since *r* can be an *n:m* relationship, the relationship *r* needs to be represented as a separate table that links *x* and *y* in the same table. This design is also known as a bridge table in data warehouse jargon. The details of the bridge table is covered in chapter 9 [103].

Table 5 - Semantic of the relationship between building elements used in defining the relevant dimension tables

Relationship	Multiplicity	Specific characteristics
BIMRL_RELGROUP	m:n	• Each object can be a member of multiple groups • Each group can have another group as a subgroup beside the ordinary elements as members
BIMRL_RELSPACEBOUNDARY	1:n	• Specific to space object only • Space to space relationship is allowed for a virtual boundary • No implied dependency, which means the element in this relationship can be removed or added without removing the actual entity and its relation
BIMRL_RELAGGREGATION	1:n	• Parts are dependent on the host • Part cannot be removed from the host and be an independent object • Exclude spatial elements
BIMRL_RELCONNECTION	m:n	• A loose relationship between objects • No dependency
BIMRL_ELEMENTDEPENDENCY	1:n	• Nested object is dependent on the host

		• Nested object in theory can become an independent object if it is removed from the host • Exclude spatial elements
Spatial Structure (BIMRL_SPATIALSTRUCTURE)	1:n	• Spatial structure is a special aggregation relationship that involves only spatial elements, i.e. IfcSite, IfcBuilding, IfcBuildingStorey, and IfcSpace • It defines a hierarchical relationship beginning with IfcSite, followed by IfcBuilding, IfcBuildingStorey, IfcSpace
Containment (CONTAINER column within BIMRL_ELEMENT)	1:n	• Object can only be placed in one spatial element as its container • The containers are only of spatial element types and therefore has a hierarchical relationship as defined in Spatial Structure

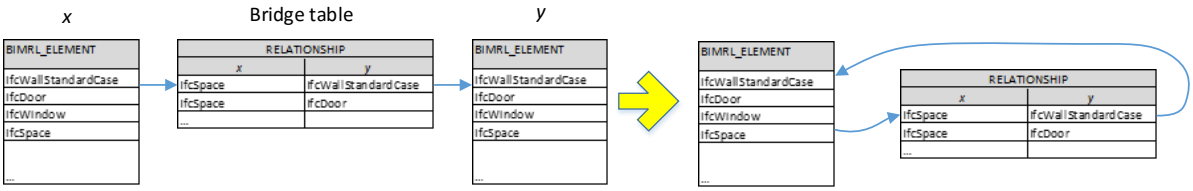


Figure 44 - The relationship table using a bridge table concept

- BIMRL_RELGROUP keeps group and membership relationships. This is used mainly for MEP system group (IfcSystem) and zone (IfcZone) that allow hierarchical groupings. No further optimization is done here to deal with the potential recursive group in this relationship since most of IfcSystem and IfcZone are one level only, with a potentially a few cases of two or three levels at most.

- BIMRL_RELSPACEBOUNDARY keeps the list of space boundary objects for a given space. The space boundary information is simplified by merging virtual and physical space boundary into one table. For the virtual boundary, we connect the space directly to the connected space in this relationship table (Figure 45).
  In BIMRL, there is an additional table that captures additional geometry and topology information of space boundaries named BIMRL_RELSPACEB_DETAIL. This table together with the original BIMRL_RELSPACEBOUNDARY are joined in a view BIMRL_SPACEBOUNDARYV that combines the information of the space boundary information and its geometry and topology information. The ETL process also enhances the space boundary information to include missing information such as virtual space boundary in some applications.

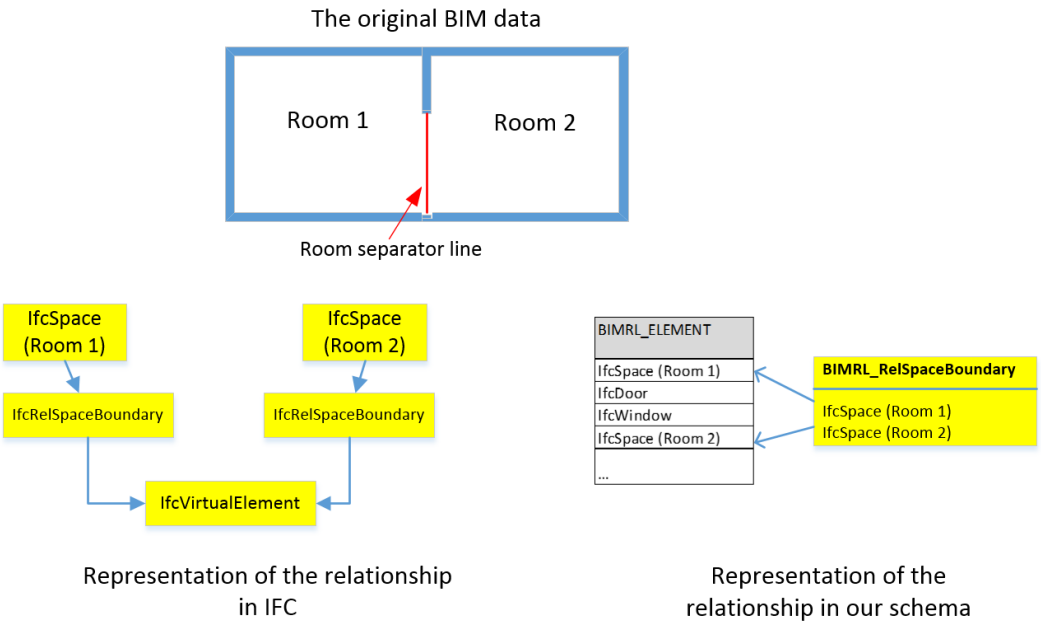


Figure 45 - Representation of the virtual space boundary relationship

- BIMRL_RELAGGREGATION keeps aggregation information. It provides a list of master and aggregated elements.
- BIMRL_RELCONNECTION contains various types of connection relationships. That includes IfcRelConnecsPathElements, IfcRelConnectsWithRealizingElements, IfcRelConnectsElements, IfcRelConnectsPorts, IfcRelCoversSpaces, IfcRelCoversBldgElements, IfcRelFlowControlElements, IfcRelReferencedInSpatialStructure, IfcRelServicesBuildings, and IfcRelConnectsStructuralElement.
- BIMRL_ELEMENTDEPENDENCY keeps a nested dependency of objects to another objects. In IFC2x3 this relationship captures IfcRelVoidsElement, IfcRelProjectsElement, and IfcRelFillsElement.

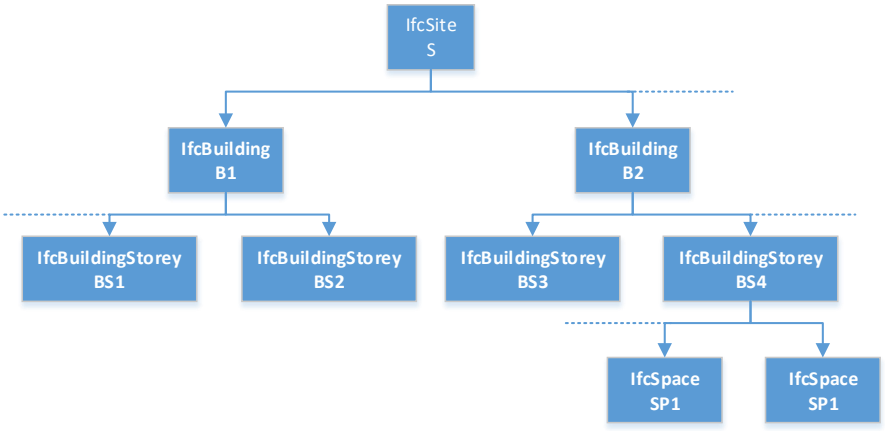
5.2.3.2 Classification

Classification information is captured in a similar manner by using bridge tables from both element (BIMRL_ELEMCLASSIFICATION) and type (BIMRL_TYPCLASSIFICATION) tables. Both point to a single non-redundant classification information in a classification table BIMRL_CLASSIFICATION.

5.2.3.3 Spatial structure

Relational databases are known to be bad in handling hierarchical or recursive relationships. Spatial structure elements in IFC is a frequently used piece of information and yet they are hierarchical. While some commercial database products provide a special way to handle hierarchical information, they are not standardized. Since spatial structure information in any building model is relatively small, the spatial structure's hierarchical structure is flattened using the hierarchy bridge table described in chapter 10 [103]. It basically creates all the relationships as an individual row in a table, marked

with information that describes how many levels removed the entities are in the hierarchy (Figure 46).



BIMRL_SPATIALSTRUCTURE		
SpatialElem	ParentElem	LevelRemoved
S	-	0
B1	S	1
B2	S	1
BS1	B1	1
BS1	S	2
BS2	B1	1
BS2	S	2
BS3	B2	1
BS3	S	2
BS4	B2	1
BS4	S	2
SP1	BS4	1
SP1	B2	2
SP1	S	3
SP2	BS4	1
SP2	B2	2
SP2	S	3

Figure 46 - A hierarchical spatial structure dimension table

5.2.3.4 Other dimensional tables

Other dimensional tables are typical dimensional tables that the fact table (BIMRL_ELEMENT) points to via foreign key. They are:

- BIMRL_MODELINFO

  It keeps the information of the individual model both as a standalone and as part of the federated model.

- BIMRL_OWNERHISTORY

  It stores the metadata information about the object. It is usually a single record in most implementations. The information is currently under-utilized and the usage is not well defined.

5.2.4 Adjacent fact table

BIMRL_TYPE table is similar to BIMRL_ELEMENT in that it has own small numbers of detailed table (BIMRL_TYPEPROPERTIES) and dimensional tables (BIMRL_TYPCLASSIFICATION, BIMRL_TYPEMATERIAL). This type of table is known as another set of fact tables adjacent to the first one.

5.2.5 Metadata

There are two metadata tables used in our system in addition to the data obtained from the building models and two views. They are:

- BIMRL_FEDERATEDMODEL

  This table keeps model information that has been populated into the database. It keeps a federated model id as a key identifier to find the relevant tables in the database for the corresponding models.

- BIMRL_OBJECTHIERARCHY

  This table is a flattened IFC entity (captured in a similar fashion as the spatial structure). The motivation to capture IFC entities and their hierarchy is to aid in a query when the query may apply to the same supertype. This allows for a shorthand in the query by just specifying the appropriate supertype. For example a query for all building elements that do not have classification:

```
select elementid, elementtype, name from BIMRL_Elemwogeom
 where elementid in (select elementid from BIMRL_Elemwogeom
      where elementtype in select elementsubtype from
      BIMRL_Objecthierarchy  where
      elementtype='IFCBUILDINGELEMENT' and
      IfcSchemaVer='IFC2X3')
minus
select elementid from BIMRL_Elemclassification);
```

Three views are defined for convenience:

- View: BIMRL_PROPERTIES

  A join table that combines properties from both an element and its type. The aggregated properties that are directly attached to the element and its type are automatically accessible in this view. i.e. for every element x that has a set of properties $P_{(x)} = \{p_1, p_2, p_3, \dots\}$ and is a type of T, $x \in T$, which in turn has a set of properties $P_{(T)} = \{p_a, p_b, p_c, \dots\}$, the total properties that are applicable to x becomes $Total\ P_{(x)} = P_{(x)} + P_{(T)}$.

- View: BIMRL_CLASSIFASSIGNMENT

  This view is very similar with BIMRL_PROPERTIES. It joins two bridge tables BIMRL_ELEMCLASSIFICATION and BIMRL_TYPCLASSIFICATION into one view complete with expanded classification information.

- View: BIMRL_SPACEBOUNDARYV

  This view is an aggregation of space boundary relationships that comes from the IFC file (IfcRelSpaceBoundary), and the enhanced data from the BIMRL ETL process to include other boundary information that sometimes is not captured in the IFC file. This enhancement is done using the geometric properties of the spaces and their boundaries.

5.2.6 Mapping of property values

Capturing IFC property values into a simple database schema poses challenges because of large number of (> 100) defined types, but more importantly types that do not hold just single values. As one main goal for the BIMRL schema is to simplify the data structure so that queries to the data is fast and efficient, the property values are all captured into a single string field. Conversion of all property single values is straightforward. However, there is a need to define a format to capture properties that are of non-single values.

- Simple Property
  - Single and enumerated value
    - String with simple conversion.
  - Bounded value
    - Stored as a pair of upper bound value and lower bound value enclosed in a square brackets and separated by a comma: **[**<*lower bound*>**,** <*upper bound*>**]**. A dash is used (‘-‘) if one value is unspecified.

- Table value
  - Stored as a list of paired values enclosed in round brackets. Each pair is separated by a comma, and each pair is separated from another pair by a semi colon:

    (<*defining value*>**,** <*defined value*>**); (**<*defining value*>**,** <*defined value*>**);** …
- List value
  - Stored in a similar format as a table value in which the individual value in the list is enclosed in round brackets and each of the value is separated with another by a semicolon:

    (<*list value 1*>**); (**<*list value 2*>**);** …
- Reference value is currently not supported because references to an IFC entity are not always well-defined in the star schema and most IFC implementations do not support it currently.

- Complex property

  The complex property is flattened by promoting the complex property into the level of property group name with extended naming. The PropertyGroupName of a complex property is always given a prefix of the property set or property name that contains it. For example:

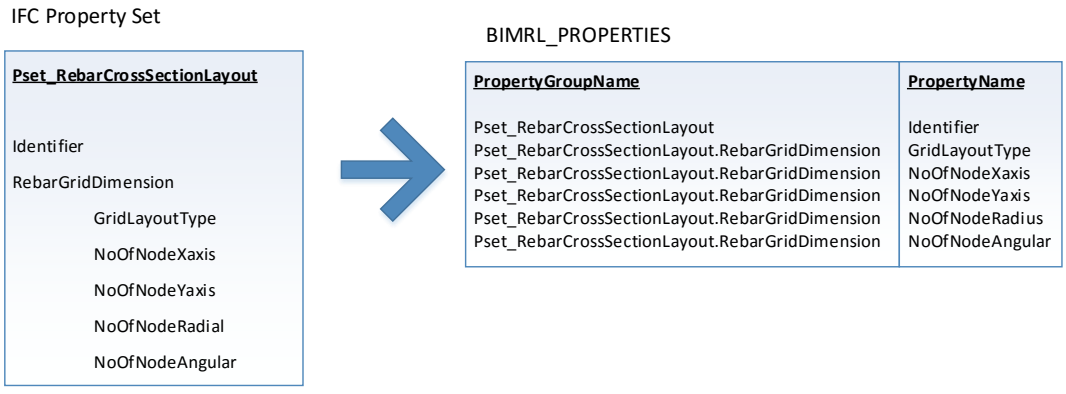


Figure 47 - Transforming IFC Complex Property Set

With these transformation rules, queries for property values can be performed easily as all values are converted to string values. The database wildcard operator LIKE can be used to query a specific value of property. For more complex properties such as bounded, table, or list values, a more complex regular expression can be used for the search condition. For example, the following statement searches for property values that has one or more table value pairs:

```
select * from bimrl_properties where regexp_like
propertyvalue,'(\([A-Za-z0-9_\.\-]+, [A-Za-z0-9_\.\-]+\);)+');
```

5.2.7 Geometry data

In BIMRL, all the geometry data is stored in the element table based on the WCS (World Coordinate System). Each geometry exists in each individual element that owns it, nothing is shared. One main reason to do this is that each instance geometry needs to be transformed to its WCS location for the correct creation of the spatial indexing. For this reason also, mapping source geometry defined for the type object and referred to using mapped item with its corresponding transformation matrix is not stored since the final geometry is already formed, indexed and stored, and hence the source does not serve any purpose.

Similarly it is applicable to the models integrated into a federated model. All the models will be in the final form that has been transformed to the WCS. The transformation matrix for each geometric structure is stored in the main table alongside the geometry data. This transformation matrix can be used to transform the model back if the need arises.

5.3 Geometry and Spatial Support for BIM database

BIM data is inseparable from its geometry and spatial location. Most building objects have geometrical shapes, which are in fact an essential part of the data that CAD

or BIM authoring tools are all about. Many queries related to the building model and building objects are spatial in nature. For example they are shown in the following examples:

- Find the location of lighting fixtures of a specific make that are due for replacement.
- Take an inventory of all medical and office equipment in a particular type of space in a hospital
- Locate gas pipes or electrical conduits embedded in a wall to ensure any work on the wall does not damage the pipe or conduit.

Support for the spatial set operations therefore is an essential part of many applications using BIM data. Eastman has identified the need for such spatial operations into an interrogation language on building data far back in 1975 [108]. Borrmann has in more recent years proposed such support to be built into BIM data using octree approximation [35-39, 109, 110], and recently with boundary representation [40]. Based on the experience with the implementation of building design code checking in Singapore (CORENET ePlanCheck), the spatial operations that are the most frequently used are the standard union, intersection, enclosure (inside or contain), and touch [44]. Unfortunately even though such techniques are well established with the 3D modeller libraries, the complexity that needs to be dealt with are for many-to-many operations in 3D space. A brute force approach generally performs badly in such scenarios and the use of a 3D modeller library is often resource intensive.

In past years, spatial support in a database has been recognized to be an important feature that is essential in the field of Geographic Information System (GIS). With the establishment of the Open GIS Consortium (OGC) in 1994 and its success in defining several standards such as simple feature, the simple feature and its spatial support gained widespread support in nearly all geospatial applications and database systems [111]. They

are currently supporting mainly 2D since maps are generally 2D. However, there is an increasing trend towards supporting 3D extensions to it for two reasons. One is that even GIS and city planning often involve 2½-D if not 3D. The second reason is that there is increasing recognition of the need to integrate GIS and building information and thus 3D support is beneficial [112]. In a similar manner, the 3D spatial support should be built into BIM database, so that integrated queries can be performed to interrogate building models for both alphanumeric data and its geometry and spatial data. This research aims to provide a proof-of-concept that such integration is feasible using a commercial database system.

5.4 Database systems supporting geometry and spatial operations

To enable integration geometry data in a database, a survey of various database management systems was undertaken to identify a suitable database that supports 3D extension to their geospatial support. At present there are only two products available and ready for use, i.e. PostGIS extension to PostgreSQL [113], and Oracle Spatial and Graph [114]. PostgreSQL with PostGIS extension has a simple and elegant 3D support that uses a simple bounding box for spatial operations on a 3D polyhedron. Unfortunately the implementation is not yet mature. The spatial indexing scheme using a simple bounding box is also too coarse for meaningful queries related to building rules. Oracle spatial on the other hand has a much more mature and stable implementation even though it is several degrees more complex than PostGIS. It also supports two step spatial evaluations, the first round using R-tree based spatial indexes to filter out all unlikely candidates, and the second for precise evaluations of the reduced candidates [114].

An ETL tool has been developed that reads an IFC file, transforms it and loads it into the Oracle Spatial database in one single step. Xbim is used as the base tool to write the extension. It provides IFC parsing capability and it processes the IFC geometries

using OpenCascade geometry modeller and generates the simplified triangulated polyhedron data. This triangulated data is the source geometry for Oracle spatial. Figure 48 illustrates the ETL process to populate the BIM data into Oracle spatial.

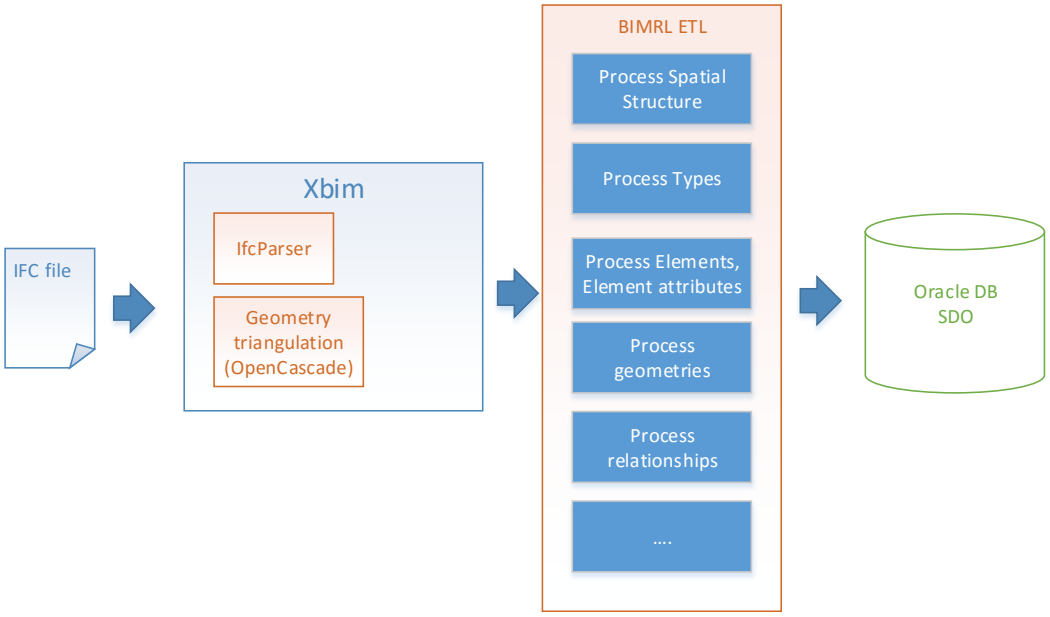


Figure 48 - Extract, Transform, Load process of IFC data into BIMRL schema

5.5 Validation Tests

The main aims with the approach for this query-able BIM are easy and efficient queries, ability to expose BIM data for ad-hoc queries and analyses, and integrated spatial support within queries. Therefore, this section demonstrates how the goals are achieved with various scenarios to query the BIM data with standard SQL using typical queries relevant to BIM. They range from relatively simple queries that make use of the capabilities of the relational database, to relatively complex queries with the integrated spatial operations within a single query expression. Three realistic building models that have large number of IFC entities and a small test model are chosen to test the queries (Model A – Model D). The statistics for the models used are shown in APPENDIX B.

5.5.1 General queries

- Query G-1:

  Generate a simple aggregate count of elements inside one federated model. The statement aggregates and counts the number of each type of element in the model, and provides quick statistics about the BIM elements inside the model.

```
select e.elementtype, count(*) from bimrl_element e
group by e.elementtype order by e.elementtype;
```

An example of a query result on Model-C is shown in Table 6 below. There are 41,913 elements in total in the model. Note that since the model is federated from seven individual models from various disciplines, there are multiple counts of IfcProject, one from each of seven individual models and one IfcProject representing the federated model. Note also that in this model the staircases are modelled as IfcStair in the Architectural model, and one additional IfcStair is an aggregate of a single IfcStairFlight modelled in the Structural model.

Table 6 - Result from query G-1

Element	Count
IFCBEAM	1970
IFCBUILDING	7
IFCBUILDINGELEMENTPROXY	3813
IFCBUILDINGSTOREY	56
IFCCOLUMN	604
IFCCOVERING	602
IFCCURTAINWALL	178
IFCDISTRIBUTIONCONTROLELEMENT	861
IFCDOOR	440
IFCFLOWCONTROLLER	583
IFCFLOWFITTING	18427
IFCFLOWSEGMENT	19352
IFCFLOWTERMINAL	4680
IFCFOOTING	553
IFCFURNISHINGELEMENT	201
IFCMEMBER	7127
IFCOPENINGELEMENT	735
IFCPLATE	2211
IFCPROJECT	8

IFCRAILING	93
IFCROOF	24
IFCSITE	7
IFCSLAB	35
IFCSPACE	193
IFCSTAIR	61
IFCSTAIRFLIGHT	1
IFCSYSTEM	790
IFCWALL	158
IFCWALLSTANDARDCASE	1310
IFCWINDOW	131

- Query G-2:

  Locate a specific type of object (e.g. Fire Extinguisher Cabinet) and show their location (which space, level they are contained in) and show the type:

```
select a.elementid as "guid", a.name as "Name", b.name as
"Container",
 f.name as "Cont Level", e.name as "Type"
from bimrl_elemwogeom a, bimrl_elemwogeom b,
bimrl_spatialstructure c, bimrl_elemwogeom f, bimrl_type e
         where b.elementid = a.container and f.elementid=c.parentid
and c.spatialelementid=a.container
and c.levelRemoved=1 and e.elementid = a.typeid
and a.elementtype='IFCBUILDINGELEMENTPROXY'
and a.name like 'Fire Extinguisher Cabinet%';
```

  The result of the above query on the Model-A is shown in Table 7 below:

Table 7 - Result from query G-2

Guid	Name	Container	Container Name	Type
2qcFpN0QH2eQhfoc89wfoJ	Fire Extinguisher Cabinet Recessed:9 1/2 x 4 1/2 x 24":2443944	114	FIRST FLOOR	9 1/2 x 4 1/2 x 24"
2qcFpN0QH2eQhfoc89wfpS	Fire Extinguisher Cabinet Recessed:9 1/2 x 4 1/2 x 24":2443947	148	FIRST FLOOR	9 1/2 x 4 1/2 x 24"
2qcFpN0QH2eQhfoc89wfp8	Fire Extinguisher Cabinet Recessed:9 1/2 x 4 1/2 x 24":2443948	129	FIRST FLOOR	9 1/2 x 4 1/2 x 24"
2qcFpN0QH2eQhfoc89wfqf	Fire Extinguisher Cabinet Recessed:9 1/2 x 4 1/2 x 24":2443941	105	FIRST FLOOR	9 1/2 x 4 1/2 x 24"
2qcFpN0QH2eQhfoc89wftz	Fire Extinguisher Cabinet Recessed:9 1/2 x 4 1/2 x 24":2443939	100	FIRST FLOOR	9 1/2 x 4 1/2 x 24"

- Query G-3:

  Find all MEP objects (IFCDISTRIBUTIONELEMENT) and Proxy objects inside a specific space. This query involves a spatial operation to ensure that all the desired objects are contained in the space. It does not rely solely on the space containment information supplied by the original IFC file. In fact, the IFC models do not contain a consistent space containment information due to the current limitation of BIM federated data currently being managed only by co-location and not being integrated [115]

  - Using Oracle Spatial R-tree index:

```
select a.elementid, a.elementtype, a.name, c.name
   from bimrl_element a, bimrl_element b, bimrl_type c
   where SDO_ANYINTERACT(a.geometrybody, b.geometrybody)='TRUE'
      and b.elementtype='IFCSPACE' and b.name='100'
      and c.elementid=a.typeid and a.elementtype in
        (select elementsubtype from bimrl_objecthierarchy
            where elementtype='IFCDISTRIBUTIONELEMENT'
            and IfcSchemaVer='IFC2X3'
         union
         select 'IFCBUILDINGELEMENTPROXY' from dual);
```

  - Using BIMRL octree based spatial index

```
select unique a.elementid, a.elementtype, a.name, c.name TypeName
   from bimrl_element a, bimrl_element b, bimrl_type c,
      bimrl_spatialindex d, bimrl_spatialindex e
   where
      d.elementid=a.elementid and e.elementid=b.elementid and
      d.cellid=e.cellid and b.elementtype='IFCSPACE'
      and b.name='100' and c.elementid=a.typeid
      and a.elementtype in
        (select elementsubtype from bimrl_objecthierarchy
         where elementtype='IFCDISTRIBUTIONELEMENT'
         and IfcSchemaVer='IFC2X3' union
         select 'IFCBUILDINGELEMENTPROXY' from dual);
```

This query applied to Model-A returns 27 IfcBuildingElementProxy and 32 IfcDistributionElement objects. The results is show in Table below.

Table 8 - Results from Query G-3 on Model-A

Element Id	Element Type	Name	Type Name
3lZPFEkJLC7BhwfhOuINeg	IFCBUILDINGELEMENTPROXY	Duplex Receptacle:Standard:775159	Standard
0WT_yZ5oz9WQHW3J8CJvaS	IFCBUILDINGELEMENTPROXY	Wall Occupancy Sensor - Dual Relay:Passive Infrared - 120 V:868521	Passive Infrared - 120 V
3lZPFEkJLC7BhwfhOuILb0	IFCBUILDINGELEMENTPROXY	Duplex Receptacle:Standard:782493	Standard
1sGszUC2r2oPqcRoWgW9Rz	IFCBUILDINGELEMENTPROXY	Junction Boxes - Load:4" Square 120 - 1:796681	4" Square 120 - 1
1shxIi5fz5G8peswD6B8tn	IFCBUILDINGELEMENTPROXY	Duplex Receptacle - Floor Mounted - Hosted - GF:Duplex Receptacle - Floor Mounted - Standard:798167	Duplex Receptacle - Floor Mounted - Standard
1bE1Vays16YOPK339jfSgZ	IFCBUILDINGELEMENTPROXY	Lighting Switches:Single Pole:863092	Single Pole
0WT_yZ5oz9WQHW3J8CJ_d_	IFCBUILDINGELEMENTPROXY	Wall Occupancy Sensor - Dual Relay:Passive Infrared - 120 V:864331	Passive Infrared - 120 V
1sGszUC2r2oPqcRoWgW9aV	IFCBUILDINGELEMENTPROXY	Junction Boxes - Load:4" Square 120 - 1:796651	4" Square 120 - 1
1bE1Vays16YOPK339jfSi7	IFCBUILDINGELEMENTPROXY	Lighting Switches:Single Pole:862928	Single Pole
3lZPFEkJLC7BhwfhOuILcL	IFCBUILDINGELEMENTPROXY	Duplex Receptacle:Standard:782408	Standard
3lZPFEkJLC7BhwfhOuINsh	IFCBUILDINGELEMENTPROXY	Duplex Receptacle:Standard:775286	Standard
1sGszUC2r2oPqcRoWgW9a1	IFCBUILDINGELEMENTPROXY	Junction Boxes - Load:4" Square 120 - 1:796661	4" Square 120 - 1
1sGszUC2r2oPqcRoWgW9aL	IFCBUILDINGELEMENTPROXY	Junction Boxes - Load:4" Square 120 - 1:796641	4" Square 120 - 1
1bE1Vays16YOPK339jfSgn	IFCBUILDINGELEMENTPROXY	Lighting Switches:Single Pole:863078	Single Pole
1bE1Vays16YOPK339jfSip	IFCBUILDINGELEMENTPROXY	Lighting Switches:Single Pole:862948	Single Pole
3Zff_ia95DJh9W$Jnf0WkP	IFCBUILDINGELEMENTPROXY	Wall Occupancy Sensor - Dual Relay:Passive Infrared - 120 V:876485	Passive Infrared - 120 V
0WT_yZ5oz9WQHW3J8CJ_R2	IFCBUILDINGELEMENTPROXY	Wall Occupancy Sensor - Dual Relay:Passive Infrared - 120 V:868215	Passive Infrared - 120 V
2qcFpN0QH2eQhfoc89wftz	IFCBUILDINGELEMENTPROXY	Fire Extinguisher Cabinet Recessed:9 1/2 x 4 1/2 x 24":2443939	9 1/2 x 4 1/2 x 24"
3lZPFEkJLC7BhwfhOuINet	IFCBUILDINGELEMENTPROXY	Duplex Receptacle:Standard:775146	Standard
1shxIi5fz5G8peswD6B8t1	IFCBUILDINGELEMENTPROXY	Duplex Receptacle - Floor Mounted - Hosted - GF:Duplex Receptacle - Floor Mounted - Standard:798183	Duplex Receptacle - Floor Mounted - Standard
3lZPFEkJLC7BhwfhOuILcT	IFCBUILDINGELEMENTPROXY	Duplex Receptacle:Standard:782400	Standard
1sGszUC2r2oPqcRoWgW9ae	IFCBUILDINGELEMENTPROXY	Junction Boxes - Load:4" Square 120 - 1:796636	4" Square 120 - 1
1sGszUC2r2oPqcRoWgW9Rw	IFCBUILDINGELEMENTPROXY	Junction Boxes - Load:4" Square 120 - 1:796686	4" Square 120 - 1
1OzBRdr1HBXg8NyQcUdeDL	IFCBUILDINGELEMENTPROXY	Thermostat:Standard:942010	Standard
1sGszUC2r2oPqcRoWgW9aI	IFCBUILDINGELEMENTPROXY	Junction Boxes - Load:4" Square 120 - 1:796646	4" Square 120 - 1
1sGszUC2r2oPqcRoWgW9a4	IFCBUILDINGELEMENTPROXY	Junction Boxes - Load:4" Square 120 - 1:796656	4" Square 120 - 1
1sGszUC2r2oPqcRoWgW9Rm	IFCBUILDINGELEMENTPROXY	Junction Boxes - Load:4" Square 120 - 1:796676	4" Square 120 - 1
0HZ6qgtUj4sPMNuJHrcWeK	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:2028544	Generic Pendant
0HZ6qgtUj4sPMNuJHrcWi$	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:2028538	Generic Pendant
0HZ6qgtUj4sPMNuJHrcWlp	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:2028542	Generic Pendant
3S9BJHH3T58hL5w6ihOkIQ	IFCFLOWTERMINAL	SE:SE Single Face Exit Sign:760517	SE Single Face Exit Sign

1bE1Vays16YOPK339jfSeH	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:863174	Generic Pendant
0HZ6qgtUj4sPMNuJHrcWmV	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:2028536	Generic Pendant
3qdkbCvcPA09WG7DgpVK$T	IFCFLOWTERMINAL	Data Outlet:Plain:815982	Plain
3S9BJHH3T58hL5w6ihOklp	IFCFLOWTERMINAL	SE:SE Single Face Exit Sign:760556	SE Single Face Exit Sign
1bE1Vays16YOPK339jfSfk	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:863161	Generic Pendant
0HZ6qgtUj4sPMNuJHrcWeN	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:2028545	Generic Pendant
3S9BJHH3T58hL5w6ihOkUY	IFCFLOWTERMINAL	SE:SE Single Face Exit Sign:760317	SE Single Face Exit Sign
1bE1Vays16YOPK339jfSt3	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:863252	Generic Pendant
1bE1Vays16YOPK339jfSej	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:863226	Generic Pendant
1bE1Vays16YOPK339jfSe4	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:863187	Generic Pendant
1bE1Vays16YOPK339jfStG	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:863239	Generic Pendant
15j$xklUz8aO4J078llzEX	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:878493	Generic Pendant
0HZ6qgtUj4sPMNuJHrcWrm	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:2028533	Generic Pendant
0HZ6qgtUj4sPMNuJHrcWlX	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:2028540	Generic Pendant
3qdkbCvcPA09WG7DgpVKcz	IFCFLOWTERMINAL	Data Outlet:Plain:816398	Plain
0HZ6qgtUj4sPMNuJHrcWiy	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:2028537	Generic Pendant
0HZ6qgtUj4sPMNuJHrcWiU	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:2028539	Generic Pendant
3S9BJHH3T58hL5w6ihOklh	IFCFLOWTERMINAL	SE:SE Single Face Exit Sign:760564	SE Single Face Exit Sign
3S9BJHH3T58hL5w6ihOklZ	IFCFLOWTERMINAL	SE:SE Single Face Exit Sign:760572	SE Single Face Exit Sign
3S9BJHH3T58hL5w6ihOkUd	IFCFLOWTERMINAL	SE:SE Single Face Exit Sign:760312	SE Single Face Exit Sign
1bE1Vays16YOPK339jfSew	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:863213	Generic Pendant
0HZ6qgtUj4sPMNuJHrcWlm	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:2028541	Generic Pendant
0HZ6qgtUj4sPMNuJHrcWmS	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:2028535	Generic Pendant
0HZ6qgtUj4sPMNuJHrcWm5	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:2028534	Generic Pendant
1bE1Vays16YOPK339jfSti	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:863291	Generic Pendant
1bE1Vays16YOPK339jfStv	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:863278	Generic Pendant
1bE1Vays16YOPK339jfSet	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:863200	Generic Pendant
1bE1Vays16YOPK339jfSts	IFCFLOWTERMINAL	Generic Pendant2:Generic Pendant:863265	Generic Pendant

5.5.2 Simple checking

The database query into the BIMRL proposed model does not only allow general queries but also enables certain types of simple checking of the data. A few scenarios are presented in this section.

- Checking query C-1:

  Check elements of IfcBuildingElementProxy type that are not assigned any classification code. The classification code may be attached to the element directly or it may also be obtained from its type.

```
select a.elementid, a.name, b.longname
from bimrl_element a, bimrl_element b
where a.elementid not in
(select elementid from bimrl_classifassignment)
and a.typeid not in
(select elementid from bimrl_classifassignment)
and a.elementtype='IFCBUILDINGELEMENTPROXY'
and b.elementid=a.container;
```

- Checking query C-2:
  - A simple query checking: Query C-2A

    Check that the classification code assigned to an element or type adheres to the correct format for OmniClass classification in form of '99-99 [99 99 99 99 99]', i.e. two digits of the chapter number, followed by one or more pairs of two digit codes up to six levels deep.

```
select a.elementid, a.elementtype, b.classificationItemCode,
b.ClassificationItemName from bimrl_element a,
bimrl_classifassignment b
where classificationname='OMNICLASS'
and (b.elementid=a.elementid or b.elementid=a.typeid)
and not regexp_like (b.classificationitemcode,'\d{2}[-
](\d{2}\s*){1,6}');
```

This query will return all rows that do not adhere to the OmniClass code format when the classification assignment is for OmniClass. An example of the query result is shown in Table 9 below.

Table 9 - Result from query C-2A

ELEMENTID	2U3XBYs756S9ILwPb1Meme
ELEMENTTYPE	IFCBUILDINGELEMENTPROXY
CLASSIFICATIONITEMCODE	A1-20 30 21 34 43
CLASSIFICATIONITEMNAME	Wrong OmniClass code

- A complex query value of property involving the use of a regular expression:

  Query C-2B

  Check if 'Level51' falls within the range of the property 'RangeSectionElevation' (of type Property Bounded Value), and get the corresponding element information if it fulfils the range conditions:

```
select a.elementid, a.elementtype, a.name,
b.propertygroupname,
b.propertyname, b.propertyvalue from bimrl_element a,
bimrl_properties b
where a.elementid=b.elementid
and b.propertyname='RangeSectionElevation'
and 'Level-51' between regexp_substr
         (b.propertyvalue,
'(\[)([A-Za-z0-9_-]+)(,\s*)([A-Za-z0-9_-]+)(\])',1,1,'i',4)
and regexp_substr
         (b.propertyvalue,
'(\[)([A-Za-z0-9_-]+)(,\s*)([A-Za-z0-9_-]+)(\])',1,1,'i',2);
```

  In this example the query found one matched entry shown in Table 10 below.

Table 10 - Result from query C-2B

ELEMENTID	2U3XBYs756S9ILwPb1MfVR
ELEMENTTYPE	IFCBUILDINGELEMENTPROXY
NAME	Family_box:Family_box:126795
PROPERTYGROUPNAME	ACIPset_RebarCrossSectionLayout_1
PROPERTYNAME	RangeSectionElevation
PROPERTYVALUE	[Level-59, Level-45]

The two regular expressions above return the fourth sub-pattern in the first regular expression, i.e. returning ‘Level-45’ (the lower bound) from the example of the result, and the second sub-pattern in the second regular expression, i.e. returning ‘Level-59’ (the upper bound). These values were checked against ‘Level-51’ in the query to see whether ‘Level-51’ falls in between ‘Level-45’ and ‘Level-59’.

- Checking query C-3:

  Check for potential issue with a space boundary, i.e. space is isolated without means to access it through a door, opening or direct connection via another space:

```
select a.elementid, a.longname, b.parentid, c.longname
from bimrl_element a, bimrl_spatialstructure b, bimrl_element c
where c.elementid=b.parentid and b.spatialelementid=a.elementid
and a.elementtype='IFCSPACE' and b.levelremoved=1 and a.elementid
not in
(select spaceelementid from bimrl_relspaceboundary
where boundaryElementType in ('IFCDOOR','IFCOPENING','IFCSPACE'));
```

  The query when applied to Model-B reports 10 spaces that do not have access to another space (isolated) as shown in Table 11. In this example, further queries reveal that two spaces highlighted in the Table 11 do not have boundary information at all, while the other 8 spaces have boundaries but without doors, openings or spaces directly connected.

Table 11 - Result from query C-3

ELEMENTID	LONGNAME	PARENTID	PARENT LONGNAME
0uB7ow60HBQB1FBV1DxGQR	Room	00Anm4s4r7luYkA9gSCCC_	FLOOR 2
3dcrQ5lTT4LuawGDbqo8oZ	Room	00Anm4s4r7luYkA9gSCCC_	FLOOR 2
1dkawGn6z3dgzZrXi7BcO$	Area	00Anm4s4r7luYkA9gSCCRV	FLOOR 1
3dcrQ5lTT4LuawGDbqo8oX	Room	00Anm4s4r7luYkA9gSCCC_	FLOOR 2
2ZbW99Y190fg_WCXyrKBJF	Room	00Anm4s4r7luYkA9gSEr4H	FLOOR 3
16HvC1ISzEneDm7CkRM53Z	Room	00Anm4s4r7luYkA9gSCCC_	FLOOR 2
2ZbW99Y190fg_WCXyrKBJ9	Room	00Anm4s4r7luYkA9gSEr4H	FLOOR 3
3u3CWVpcj8aAgOJhGew7PX	Conf	1KDsPJhOj48Q_lsUui8flb	SECOND FLOOR
2ZbW99Y190fg_WCXyrKBJ5	Room	00Anm4s4r7luYkA9gSEr4H	FLOOR 3
1u2ogypu56dPMrJgq97XhE	Room	00Anm4s4r7luYkA9gSCCRV	FLOOR 1

- Checking query C-4:

  List specific furniture elements and equipment inside a specific space for compliance to a space program. All the objects must be taken into account regardless whether they are contained in space or not. This is achieved by making use of the 3D spatial index to find all elements that are related to the space.

  In the following example applied to MODEL-A, we searched for a Projection Screen and Data Outlet terminals inside the room.

  - Using Oracle Spatial R-tree index:

```
select unique c.elementid, c.elementtype, a.ifctype, a.name,
   a.tag, a.predefinedtype, c.name, c.modelid
from bimrl_type a, bimrl_element c, bimrl_element d
where a.elementid(+)=c.typeid
   and d.elementtype='IFCSPACE' and d.name='106'
   and SDO_ANYINTERACT(c.geometrybody, d.geometrybody)='TRUE'
   and (upper(c.name) like 'DATA OUTLET%'
   or upper(c.name) like 'PROJECTION SCREEN%');
```

  - Using BIMRL octree based spatial index:

```
select unique c.elementid, c.elementtype, a.ifctype, a.name,
   a.tag, a.predefinedtype, c.name, c.modelid
from bimrl_type a, bimrl_element c, bimrl_element d,
   bimrl_spatialindex e, bimrl_spatialindex f
where a.elementid(+)=c.typeid and d.elementtype='IFCSPACE'
   and d.name='106' and e.elementid=c.elementid
   and f.elementid=d.elementid and e.cellid=f.cellid
   and (upper(c.name) like 'DATA OUTLET%'
   or upper(c.name) like 'PROJECTION SCREEN%');
```

- Checking query C-5:

  Find any Receptacle/electrical socket that is located too near (within 1m (1000mm)) to a wet sink and therefore is in danger of causing a short circuit.

```
select a.elementid, a.elementtype, a.name, b.elementid,
b.elementtype, b.name
from bimrl_element a, bimrl_element b
where a.elementid='1pl7thiAL90hISwq2I9x0O'
and b.elementtype='IFCBUILDINGELEMENTPROXY'
and upper(b.name) like '%RECEPTACLE%'
and sdo_within_distance(b.geometrybody, a.geometrybody,
'distance=1000')='TRUE';
```

5.5.3 Integration to the rest of the enterprise data

One of the benefits of importing BIM data into a relational database is the possibility to integrate data and query to the rest of the enterprise data, which is most likely stored in a relational database. In this scenario we present one possibility of integrating such queries with the BIM data.

- Query I-1:

  Report all equipment of a certain type, which have had an associated service incident report in the past 2 months, and their location.

```
select a.elementid, a.elementtype, a.name, b.propertyvalue
"MANUFACTURER",
c.longname as "LOCATION", c.name as "CONT. LEVEL",
e.reportedby, e.reporteddate, e.servicedby, e.servicecompletion
from bimrl_element a, bimrl_properties b, bimrl_element c,
bimrl_spatialstructure d, ServiceRequest e
where propertygroupname='Pset_ManufacturerTypeInformation'
and propertyname='Manufacturer' and b.elementid=a.elementid
and d.spatialelementid=a.container and d.levelremoved=1
and c.elementid=a.container and e.elementid=a.elementid
and a.elementtype='IFCFLOWTERMINAL' and upper(a.name) like
'%LIGHT%'
and e.reporteddate > sysdate-60;
```

  In this example, a query returns results shown in Table 12 below combining information from BIM, its spatial location and the service request information from an external table outside of BIM data.

Table 12 - Result from query I-1

ELEMENTID	3yV74NdEz09vjHyfVxYAYl	3yV74NdEz09vjHyfVxYApc
ELEMENTTYPE	IFCFLOWTERMINAL	IFCFLOWTERMINAL
NAME	Pendant Light - Linear - 2 Lamp - with Emergency:48" - 120V - 3 Lamp:750448	Recessed Can Light - Non-hosted:Open Downlight:751417
MANUFACTURER	Alera	CON-TECH
LOCATION	Arch-BEARING	MP Room Lighting
CONTAINER LEVEL	Arch-BEARING	MP Room Lighting
REPORTEDBY	User 2	User 3
REPORTEDDATE	22/9/2014	25/9/2014
SERVICEDBY	Service Engineer 1	Service Engineer 1
SERVICECOMPLETION	27/10/2014	27/10/2014

5.5.4 Performance measurement

To get an idea of how efficient queries to the BIM data are using this schema, simple measurements are done using Oracle SQL Developer UI to perform the queries interactively. Table 13 below shows the records of the measurements.

Table 13 - Runtime performance measurement

Query	Model	Time (s)	Time (s) using BIMRL spatial index
G-1	Model-A	0.012	N/A
	Model-B	0.014	N/A
	Model-C	0.020	N/A
G-2	Model-A	0.087	N/A
	Model-B	0.288	N/A
	Model-C	0.893	N/A
G-3	Model-A (name='100'	66.36	8.304
	Model-B (name='156')	500.323	2.139
	Model C (name='25')	75.512	2.605
C-1	Model-A	0.032	N/A
	Model-B	0.069	N/A
	Model-C	0.185	N/A
C-2A	Model-A	0.090	N/A
	Model-B	0.137	N/A
	Model-C	0.477	N/A
C-2B	Model-A	0.362	N/A
	Model-B	0.146	N/A
	Model-C	0.056	N/A
C-3	Model-A	0.018	N/A
	Model-B	0.024	N/A
	Model-C	0.025	N/A
C-4	Model-A	17.504	0.67
C-5	Model-D	0.599	N/A
I-1	Model-A	0.102	N/A

Most queries are completed in a fraction of a second, with the exception of the queries using spatial operations making use of the R-tree indexing in Oracle Spatial (G-3 and C4). The performance with spatial operations depends heavily on how big the search

set and the result set are. In these queries, they return relatively large numbers of objects from a relatively large sub-set of data compared to the other scenarios.

5.6 Limitations

The BIMRL schema is not intended to solve every problem in the BIM world, and neither there should be a single solution to every problem. The first and foremost limitation in this approach is that the transformation of BIM data into our schema is one-way. That is only practical if we view the BIM data as read-only access. It does not exclude modification to the data, but it is not designed to easily bring back the modification to the original BIM authoring tool.

Another limitation is the exclusion of the process data from the current ETL process. It is not about the limitation of the approach, it is rather just a means of limiting the scope of the research. One main consideration for the exclusion is that the process information is still very scarce and there is not yet a standard agreement of how the IFC schema should be used for the process related information. The best methodology to extend this schema for the process related data is to extend the schema and create a new data-mart based on the process information. This is feasible as the process information is relatively separate from the building model. This new data-mart is then linked to this schema as a related data-mart, typically via a link table to pair the keys from the two fact tables. Integration and access to all the data are automatically taken care of with the same SQL interface. Figure 49 below illustrates the possible way of extending the schema to include support for process related data in future.

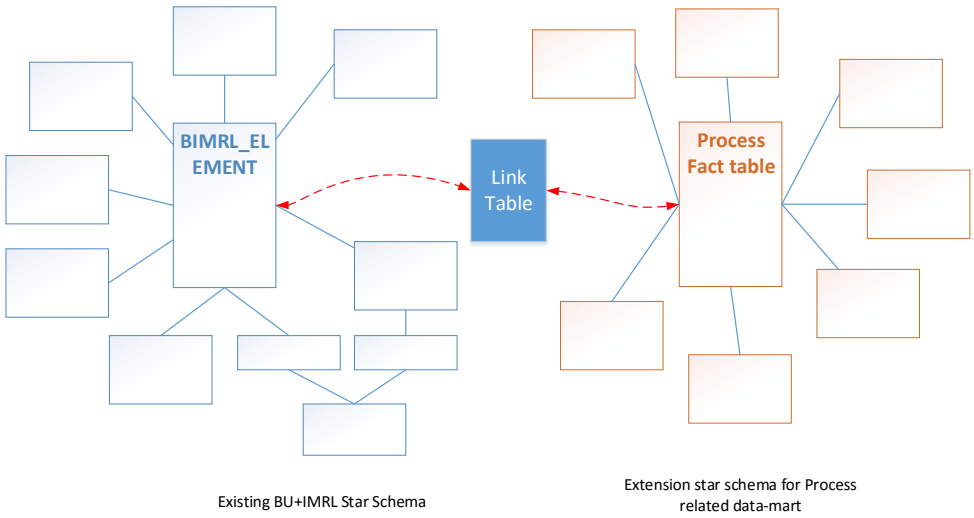


Figure 49 - Possible extension to support process related data

Performance in general is excellent despite the large models used for the validation tests (the three main models have more than 1 million IFC entities). Slower performance is expected for any query involving spatial operations. This is reasonable given how much more complex it is to evaluate the 3D geometries. It needs a better and a more mature 3D geometry and spatial support that will overcome the limitations of the current Oracle spatial data object (SDO).

5.7 Conclusions

In this chapter the feasibility of an easy, efficient, and fast query-able building model with built-in spatial support has been defined and validated. The initial test cases have shown the power of this schema and the wide-range potential for use as a decision support system allowing many types of queries beyond those typically are pre-defined. With spatially enabled database the potential is even more limitless, enabling insight into BIM data that was not possible before without a significant amount of effort. One most important benefit of this approach is the use of standard SQL to access the data. It enables ad-hoc queries to be performed on the BIM data with a knowledge of standard SQL, plus a familiarity with a simple and intuitive schema, and a familiarity with the

SQL extension for the 3D spatial and graph support. It also provides benefits of integration with the rest of the enterprise data.

To overcome the limitation of the built-in R-tree spatial indexing in Oracle Spatial, an alternative octree base spatial indexing is used for BIMRL rule checking system. This is part of an approach using multiple representations. This approach is described in the following chapter.

CHAPTER 6

MULTIPLE REPRESENTATIONS FOR BIM GEOMETRY DATA

3D geometry modeling has been an established subject for more than 30 years. It has matured in the mechanical and manufacturing domains and for the past few years it has been maturing in the BIM domain. Much of it is still locked in a proprietary format in each of the BIM authoring or CAD applications. Exchange of the intelligent information between applications is still limited beyond the geometry and visualization. While there several exchange formats for geometry and visualization purposes such as SAT, FBX, COLLADA, etc., IFC is the only viable choice for the intelligent information exchange among various BIM authoring tools. For analysis of the BIM data, efficient storage and access to the data including geometry and the relevant operations are needed. Unfortunately, this area is still lacking. There are several components that are needed to achieve this, i.e. the appropriate storage that allows alpha-numeric data, geometry data and the relationship between the two to be stored, the standardized spatial algebra that provides efficient access to the geometry and spatial data, and the standardized query language that is needed to provide the access. Currently, almost none of them have been effectively solved for BIM specific data. There has been some progress made in the area of database storage of the IFC data, for example with commercial specialized database management systems such as EDM [100], or open source BIMserver [41]. However, they are generalized database management systems that are capable of storing and managing IFC data but do not deal much with the geometry beyond its visualization.

Much development for support of a standardized storage and access to spatial data in the database management system occurred in the GIS domain. OGC's simple feature specifications suitable for storing and accessing GIS related data was developed since 1997 and version 1.1 of the specifications was first published in 2005. Various RDBMS

vendors developed their supports for geographical data at the same time. One notable support by a major vendor was by Oracle in 1996, called Spatial Data Option (SDO) with its 7.3.3 release, prior to the OGC Simple Feature specifications. Today, major RDBMS products have native support for OGC Simple Feature directly inside the database. However, they mainly deal with 2D geometry that is commonly used in the area of GIS. Several researchers have looked into extending the support to 3D in a database management system [112, 116]. While the fundamentals for geometry and topology have been well established since the 1980s and the benefits for use in various applications in the geospatial world and BIM identified, including rule checking, the progress for an integrated 3D support in a database management system has not kept up with it [117]. Most CAD or BIM application works backward, i.e. maintaining its own database system and providing a link with the external database system for other information. From the data integrity perspective, it often poses issues of inconsistency between the two databases. Also from the perspective of generalized query, there is very little that can be done to perform an integrated query that works for both geometry, topology and the alphanumeric data easily.

With the wider adoption of geospatial data in mainstream computing and wider usage of such data for day-to-day applications such as maps, there is a growing support for storing 3D data into database management systems. There are currently two RDBMS that support simple storage and queries on spatial data with extension to 3D. They are Oracle Spatial and Postgress with PostGIS extension. In both cases, the 3D geometry data is stored as a simple Polyhedron. Spatial indexing is supported in Oracle Spatial with R-tree indexing, while PostGIS uses simple axis-aligned bounding boxes (AABB). Oracle Spatial also supports storage of topology data [118]. In this research, Oracle Spatial is selected as it is a more mature implementation and with more spatial operators that can be applied to 3D objects. However, further along in the research, even with Oracle Spatial implementation, the support for spatial indexing, spatial query and query

performance on 3D geometries is still very limited. Most operations involving one-to-many spatial interactions are still very slow in the order of many seconds (refer to the test result in chapter 5.5.4). This observation is supported by research reports in [119]. This renders the system to be rather impractical for many BIM rules that often requires performing many-to-many spatial evaluations throughout the entire building model that is relatively dense compared to the typical geo-spatial data such as the city data used in [119].

Part of this research hence focuses on optimizing BIM data storage for both alphanumeric data (CHAPTER 5) and geometry data. The immediate interest is to be able to support efficient queries to a BIM model with a large variety of rules that perform well in many-to-many spatial relationship without being severely impacted by how many objects are involved in the queries.

6.1 Dealing with 3D geometry

When dealing with a 3D geometric object, there are several subject matters that are worth considering. One main question is what exactly we try to model. Most of the 3D geometry modelling subjects have been well researched. The issue of mapping the real 3D physical object into a computer model has been largely resolved by various schemes, and the boundary representation is the most popular one in use today [120-123]. The mathematical model for such geometric representation is robust since it can be validated rather easily and consistently as they obey the Euler operators [124, 125]. Beside the boundary representation there are various other schemes that are also used for different purposes, for example CSG, approximation to the shape such as subdivision of object space (of which Octree is one), and triangulated mesh that are important for visualization [126]. The existing geometric modeling has served well to support the advanced CAD and BIM applications we see today. They mainly deal with the generation of the object and maintaining the integrity of the model as they are being created and modified. The

same model though is not easy to be efficiently stored, retrieved and queried for various interrogations into its spatial relationship between unrelated objects in the model space outside of its originating application.

One aspect that rule checking requires is to be able to query and interrogate the model for spatial relationship between objects and their properties. It calls for a uniform storage of the all types of data: geometry, topology, spatial, relationship and properties, in one location. Data model such as IFC offers to capture most of them, but it is not optimized for storage, retrieval and query. The database management system with support for 3D geometry seems to be a perfect place where all the aspects of data can be stored [112]. The challenge is no longer about how to precisely model the 3D objects and maintain their integrity, but rather it is about what appropriate representations will be suitable for efficient storage, retrieval and query. Highly structured geometry and topology becomes hard to interrogate and heavy in term of computing cost, and not suitable for operations involving a large number of objects in a query.

The answer to the above seems to be a trade-off between accuracy, storage space and performance. Since rule checking involves mainly read-only access to the data and requires efficient access to query the data for properties, relationship, geometry and spatial operations, there is no one representation that can satisfy all. Therefore, in this research, multiple representations of the geometry and topology is proposed. The details of these will be covered in the rest of the chapter.

6.2 Multiple representations

Multiple representations become necessary as different operations require different representations to work efficiently. The following examples highlight the requirements for multiple representations. The number in a circle is a reference to the specific representation type as shown in the corresponding circled number in Figure 50.

- *"Identify the external face of an external wall and calculate the surface area excluding the openings"*, requires a wall surface information minus all the openings. This information can be obtained from the Brep Face ①.

- *"Define a line of sight from a Nurse Station to the door opening of Patient Room"*. This information requires a space boundary connection information to identify positions at both end of the line that share a common circulation space ①.

- *"Find all objects that block a line of sight"*. This requires an exact intersection operation between a line and a limited set of objects (polyhedra) ②

- *"Find all objects that are within a region of interest"*. For fast query, approximate shape of objects are needed to perform this type of spatial query ③.

- *"Identify objects that are below another object"*. This is another example that needs a spatial query using the spatial index ③.

- *"Identify bounding box footprint of an irregular shaped object in a space"*. This query requires a simplified form of objects in its oriented bounding box that will provide a bottom face that represents the footprint ④ or ⑤ in some cases.

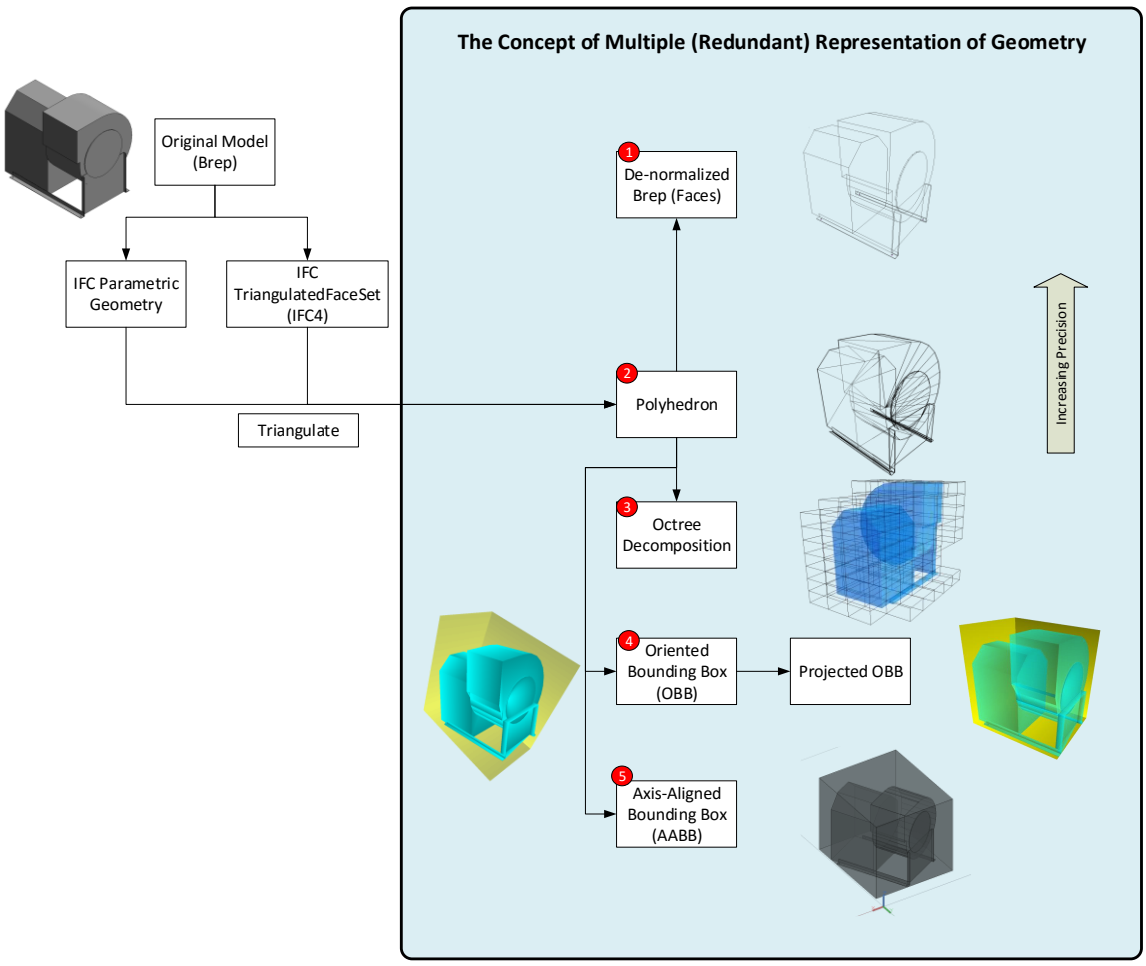


Figure 50 - The concept of Multiple Representation for Geometry Data

6.2.1 Polyhedron as the main geometry data type

The first representation is in regards to the main storage of the geometry. For this, the option is limited. Since the RDBMS supports 3D geometry in form of polyhedron boundary representation, this will be the main form of the storage of the geometry. Oracle Spatial supports other geometry types, i.e. Point and Multipoint, Line and Multiline, Polygon and Multipolygon, solid and Multisolid, but in this research the focus is mainly on the 3D solid to capture the BIM data, which is mainly solid. Polygon and line are also used as complimentary information to the main object. Details of the usage of other geometry types will be described in more detail in the subsequent sections when the appropriate topic is discussed.

In this research, the geometry data is processed through Xbim to obtain the triangulated polyhedron. The main considerations of this approach are:

- Triangulated mesh or polyhedron is the most widely available geometry data since it is used for visualization purposes. Therefore, this type of geometry can be obtained even if the source is not IFC, for example Autodesk's DWF, X3D, etc. In IFC4, a new geometry type IfcTriangulatedFaceSet has also been introduced and it is the main part of the Reference View MVD [127]. Autodesk Revit has built in support for exporting the geometry in this MVD since its 2015 version. As this option was not available when this research started, an additional step is needed to convert the parametric geometry types in IFC to the triangulated polyhedron via Xbim in IFC2x3 format. Currently, the geometry is assumed to be solid since in many cases, it is important to be able to deal with the "intersection" of an object that is completely inside another object. It is very straightforward to support a shell geometry (a hollow geometry) if required. This will be discussed in the Spatial Indexing section below.
- Triangulated geometry means a slight loss of precision due to the approximate nature of the triangulation. The triangulation also means that the boundary representation surface that can be derived from a polyhedron is limited to planar surfaces. In BIM rule checking this impact is acceptable since we are not dealing with high level precision. The same cannot be said if the geometry is required for precise fabrication.

6.2.1.1 Oracle SDO

Oracle SDO uses a combination of the User Defined Type (UDT), and stored procedures in a schema called MDSYS that supports storage and functions to operate on the geometry. The structure of SDO_GEOMETRY type is shown below:

```
CREATE TYPE sdo_geometry AS OBJECT (
        SDO_GTYPE NUMBER,
        SDO_SRID NUMBER,
        SDO_POINT SDO_POINT_TYPE,
        SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY,
        SDO_ORDINATES SDO_ORDINATE_ARRAY);

 CREATE TYPE sdo_point_type AS OBJECT (
        X NUMBER,
        Y NUMBER,
        Z NUMBER);
        CREATE TYPE sdo_elem_info_array AS VARRAY (1048576) of NUMBER;
```

CREATE TYPE sdo_ordinate_array AS VARRAY (1048576) of NUMBER;

Oracle SDO stores the geometry in an array of double (for points) in SDO_ORDINATES type. The array is a simple forward sequential list of ordinates that stores vertex information. SDO_ELEM_INFO type keeps the pointer references to the array that indicates the boundary of each element of the geometry. It essentially contains a list of elements of the geometry with the appropriate start offset and length from the starting ordinate to the last ordinate in the sequential list of ordinates in SDO_ORDINATES (Figure 51). SDO_GEOMETRY object relational entity provides a set of functions that can operate on the geometry. In addition, there are Oracle PL/SQL packages that provide more geometry and spatial functionality to SDO_GEOMETRY data. The functions are mainly designed to work with 2D geometry. Only a limited number of functions work with 3D geometry.

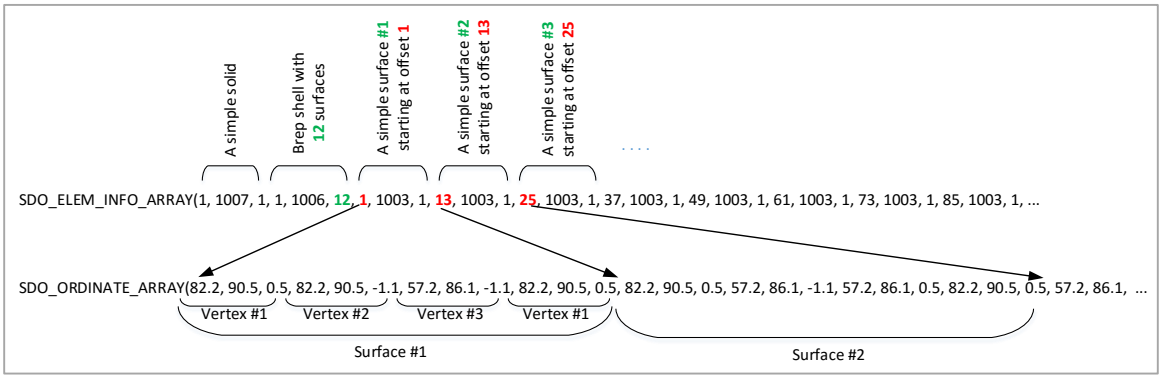


Figure 51 - SDO_GEOMETRY Organization

6.2.2 Indexed boundary faces

While the triangulated polyhedron is a valid boundary representation model, it loses the topological information since it is a de-normalized form of boundary representation. In this representation, the triangles are usually unordered and do not have shared edges as expected in the proper boundary representation model. In many cases, BIM rule checking requires more information than just the solid, such as faces and the holes or openings. For example in checking openings of an external wall/façade, one will require the face information and its openings (Figure 55). To address this issue, a step within the ETL processes the polyhedron data and "stitches" the triangles back into their original faces. In this process, if there are holes in the original surface, it will be restored (Figure 52). The fundamental support of the reliability of such an approach can be described as follows:

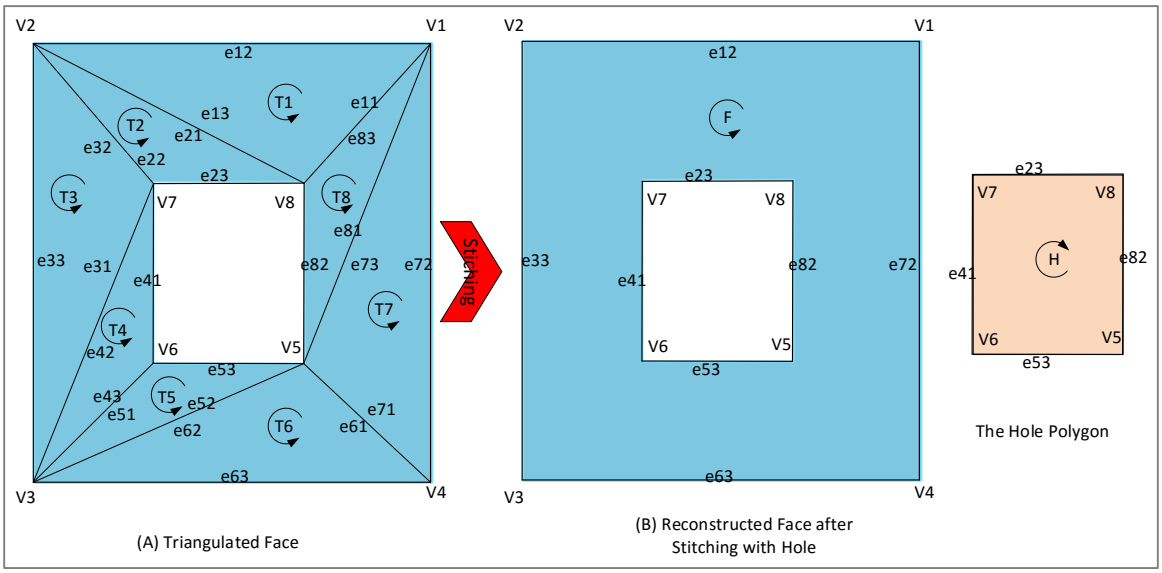


Figure 52 - Reconstructing the Original Face from the Triangulated Faces

For every solid φ in the 3D world space $\mathbb{R}^3$ that has been triangulated, there will be two types of edges, i.e. true edges that bind two faces, and "artificial" edges that are created by the triangulation process. Following Euler's equation, each of the edges ε will

bind exactly two faces $F_i \; and \; F_j, where \; i \neq j$, regardless of the type. In a system where topology is supported, the data structure usually involves Solid – Faces – Edges – Vertices (Figure 53). In this research, the BIM geometry data supported is the simplified triangulation where the topology information is not preserved explicitly, i.e. Solid φ is a set of triangulated faces F_i: $\varphi = \phi\{\mathrm{F}_i\}$, in turn $\mathrm{F}_i = \phi\{\varepsilon_j\}, \varepsilon_j = \phi\{v_k\}$, where are ε_j edges and v_k are vertices. No information is explicitly shared between those sets, they implicitly contain the "shared" information. General use of such geometric sets in the viewer is completely adequate, since the graphics card works with triangles. It is also adequate for simple geometry use such as collision detection that works by examining triangle-triangle intersection. For rule checking use this information is inadequate. As mentioned above, some questions such as those that require the knowledge of a complete face of a geometry cannot be satisfied easily with just a soup of unorganized triangles. One example is the rule found in Singapore Fire Code 2013 – clause 2.5.4(b) requires more precise information of a face of a façade and holes on it (Figure 55). There are many references to such requirements in the building codes and it will be expensive to derive such information by computing it on demand. Therefore, in this research, such topological faces have been identified as one important piece of information that needs to be derived using the strategy: compute once and read many. In almost all cases known, the topological information required is down to the Surface level. Therefore storing the de-normalized data to the surface level is sufficient. The edges and vertices are not required to be stored separately and can be derived during runtime as necessary.

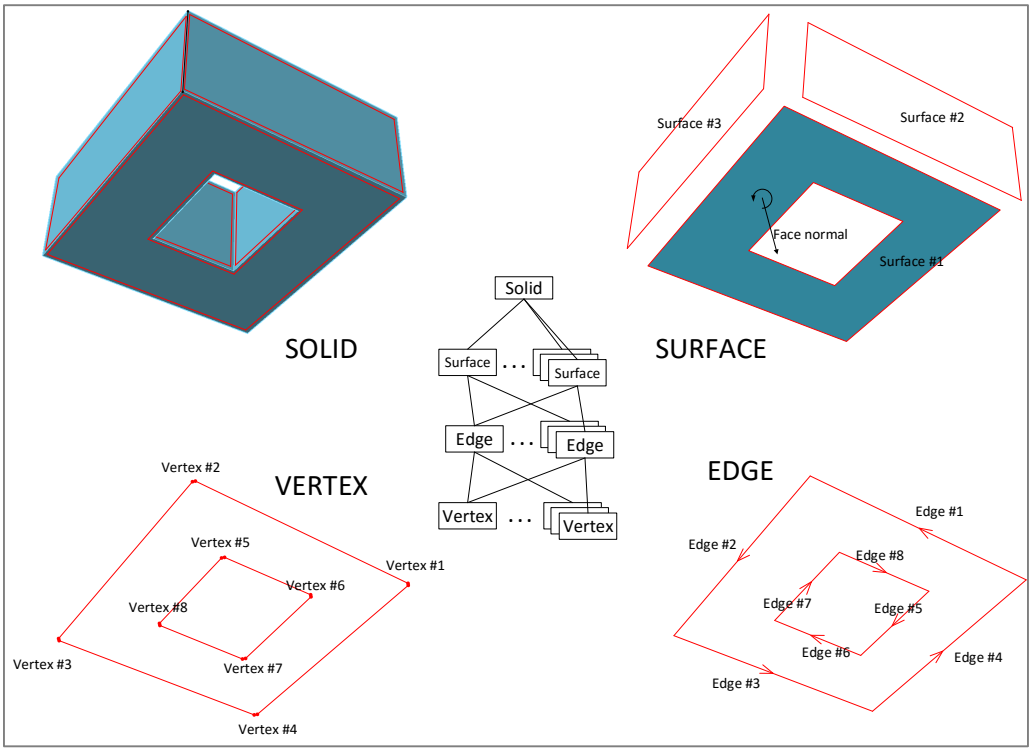


Figure 53 - Typical Topological Data Structure in 3D Solids

To be able to derive the faces, the de-normalized data in the separate sets must be re-organized and reversed-engineered. Using Euler's principal, we know that each edge binds exactly two faces. For the faces that are directly representing the solid, the face normals are usually different for each face, but for an artificial edge, the faces the edge belongs to share the same normal for a planar surface (Figure 54). By sorting the normal, the planar faces can be re-constructed because each of the shared edges (usually in the reverse direction) that belong to faces that share the same normals can be merged. This process is also known as stitching. When stitching is complete, any edges that are not merged will be the true edges that are usually called co-edges. These share the same edge with another boundary face of the solid. A hole can be identified when there is a disjoint in the vertex list and the direction of the edges that belong to the hole is expected to be in a reversed direction (Figure 52). The algorithm to stitch the triangulated faces is as follows:

```
For each solid, sort faces by vertices
For each vertex, group faces that share the same normal
   For each group of normal
      Evaluate edges from the member faces of the group
      Merge 2 faces when there are 2 edges that share the same
      vertices and having the opposite direction (generally),
      reverse it when it is not
      Update the solid set replacing the merged faces with the
      new one
      Repeat the process, until there is no more face in the
      group
For each completed face (there maybe multiple faces for the same
normal), identify the outer edges and the holes
Each face is assigned an ID, likewise each of the hole
```

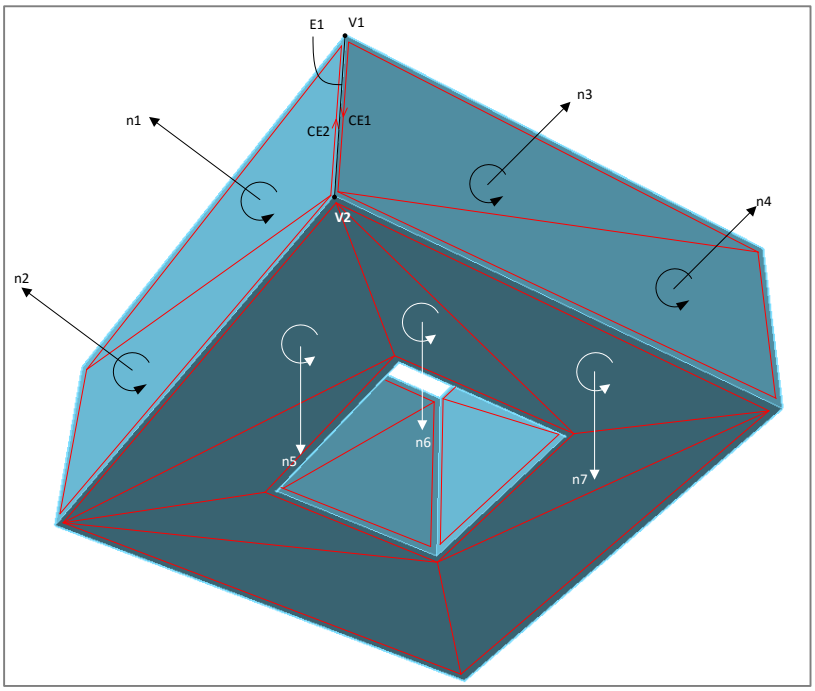


Figure 54 - Boundary Representation with Triangulated Faces and Their Normal Vectors

Each of the indexed faces is also evaluated for its orientation. Table 14 lists the predefined types of orientation (see also Figure 56).

Table 14 - Predefined Face Orientation of a Face

Face orientation	Criteria
TOP	is set for a face F if its normal $n_F = \{0.0,0.0,+1.0\}$ within $\pm$10% tolerance and one of its vertices v_{max} is the highest in V_F, where $v_{max} \in V_F$
TOPSIDE	is set for a face F if one of its vertices v_{max} is the highest in V_F, where $v_{max} \in V_F$, and the angle of the normal to +Z-axis $-\frac{1}{4}\pi < \theta(n_F) < \frac{1}{4}\pi$.
BOTTOM	is set for a face F if its normal $n_F = \{0.0,0.0,-1.0\}$ within $\pm$10% tolerance and one of its vertices v_{min} is the lowest in V_F, where $v_{min} \in V_F$
UNDERSIDE	is set for a face F if one of its vertices v_{min} is the lowest in V_F, where $v_{min} \in V_F$, and the angle of the normal to -Z-axis $-\frac{1}{4}\pi < \theta(n_F) < \frac{1}{4}\pi$.
SIDE	is set for a face F if its normal $Z(n_F) = \{0.0\}$ within $\pm$10% tolerance. The SIDE faces combined with the information of its angle from the True North will determine the exact orientation

A special face for a hole is also created as a separate face in addition to the main body and labeled as "HOLE" with the same normal as the main body of the face (opposite direction of the actual hole in the main body of the face). This is to enable direct spatial relations to be evaluated also with the hole.

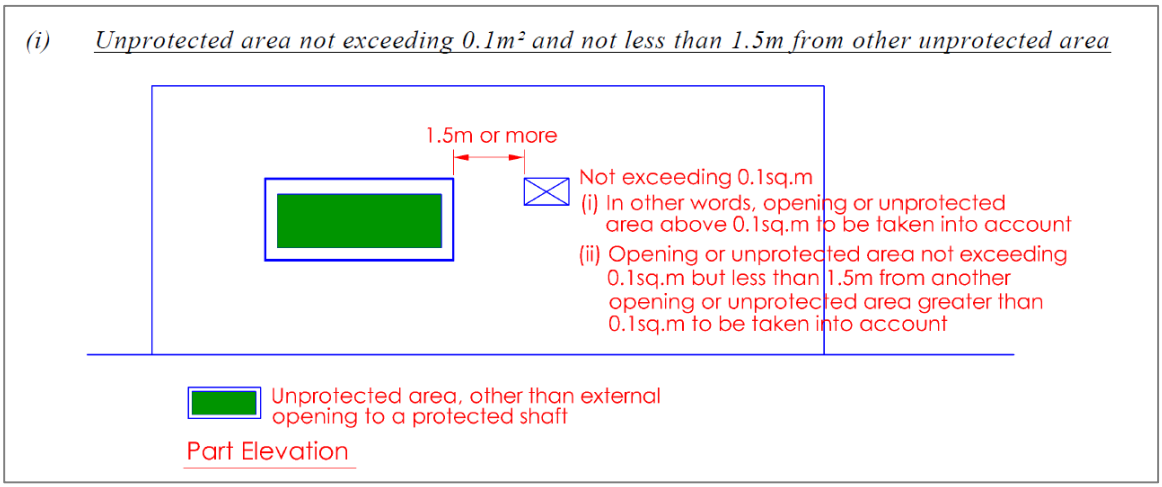


Figure 55 - An Example of Rule that Requires the Complete Face and Opening(s) on It (Singapore Fire Code 2013 – 3.5.4(b))

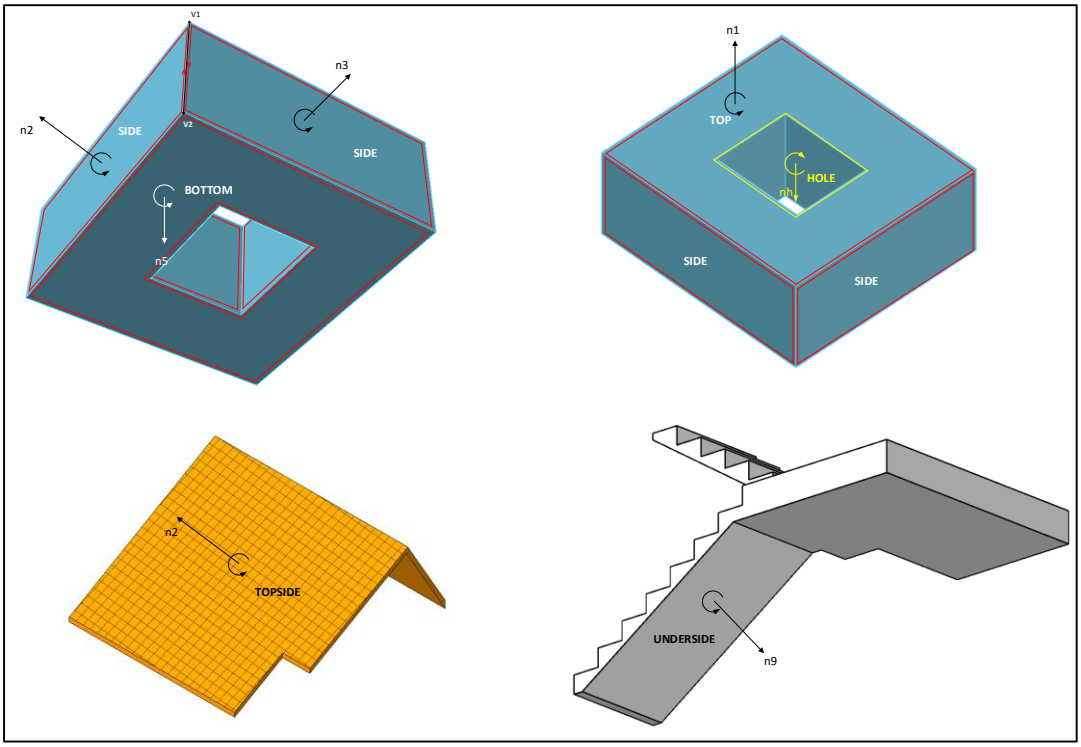


Figure 56 - Indexed Faces with Orientation

6.2.3 Oriented Bounding Box

For an irregularly shaped object, it is often useful to use its approximate shape using its bounding box. There are two types of bounding boxes, i.e. AABB (Axis-Aligned Bounding Box) and OBB (Optimized or Oriented Bounding Box). AABB has advantage of being very easy to compute and to compare, but it does not behave very well for certain geometries that are not axis-aligned, especially when it is elongated like a pipe or a duct (Figure 57). Both AABB and OBB are processed when a geometry is processed for its spatial index.

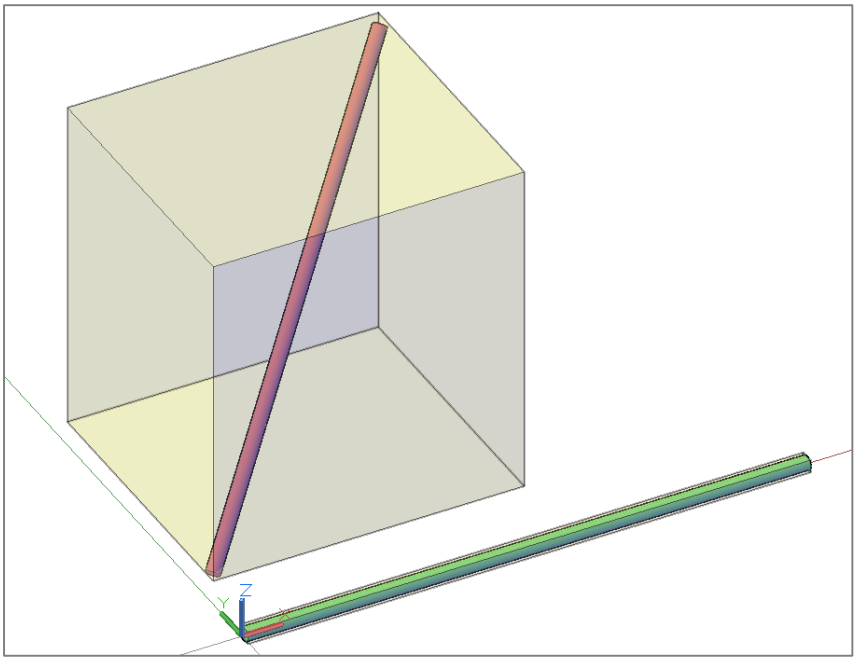


Figure 57 - Axis-aligned Bounding Box of Two Similar Objects in Different Orientations

The information from both is stored in the database for appropriate use in a query or rule evaluation. AABB can be computed with a very straightforward calculation of the maximum and minimum of X, Y, Z for each of the vertices. OBB on the other hand is ambiguous. It can be computed using a technique called PCA (Principal Component Analysis) that uses statistical data and Eigen values by analyzing a set of points. It works well for a point cloud data and it can also be used for BIM data even though the result is not always very good due to the limited number of the point set (Figure 58). The result of PCA does not always align to the axis as shown in Figure 58 (B) and (D). Since many building objects are Z-axis aligned (vertical), another form of OBB is also created with adjusted alignment to the Z-axis so that the OBB is projected to the X-Y plane, resulting in well-defined TOP and BOTTOM faces. The algorithm for OBB generation works as follows:

```
Collect point set for a geometry
Compute the centroid by computing the mean. This will be the
origin of the PCA axes.
```

Apply PCA to the point set. Using statistically significance by comparing the standard deviation, the major axes can be identified using their Eigen value. The Axis where the minimum standard deviation is the main axis, followed by the other two axes. They are also known as Eigen vectors.
Define transformation matrix from the Eigen vectors
Transform the points to the new coordinate system
Calculate the bounding box
Transform back the 8 vertices that belong to the bounding box to the world coordinate system. This will be the OBB.

For the Z-aligned OBB, extra steps are added into the algorithm to modify the OBB's Z-axis to the world Z-axis:

Collect point set for a geometry
Compute the centroid by computing the mean. This will be the origin of the PCA axes.
Apply PCA to the point set. Using statistically significance by comparing the standard deviation, the major axes can be identified using their Eigen value. The Axis where the minimum standard deviation is the main axis, followed by the other two axes. They are also known as Eigen vectors.
Adjust the axes by adjusting the Z-axis to the world Z-axis
Define transformation matrix from the Eigen vectors
Transform the points to the new coordinate system
Calculate the bounding box
Transform back the 8 vertices that belong to the bounding box to the world coordinate system. This will be the OBB.

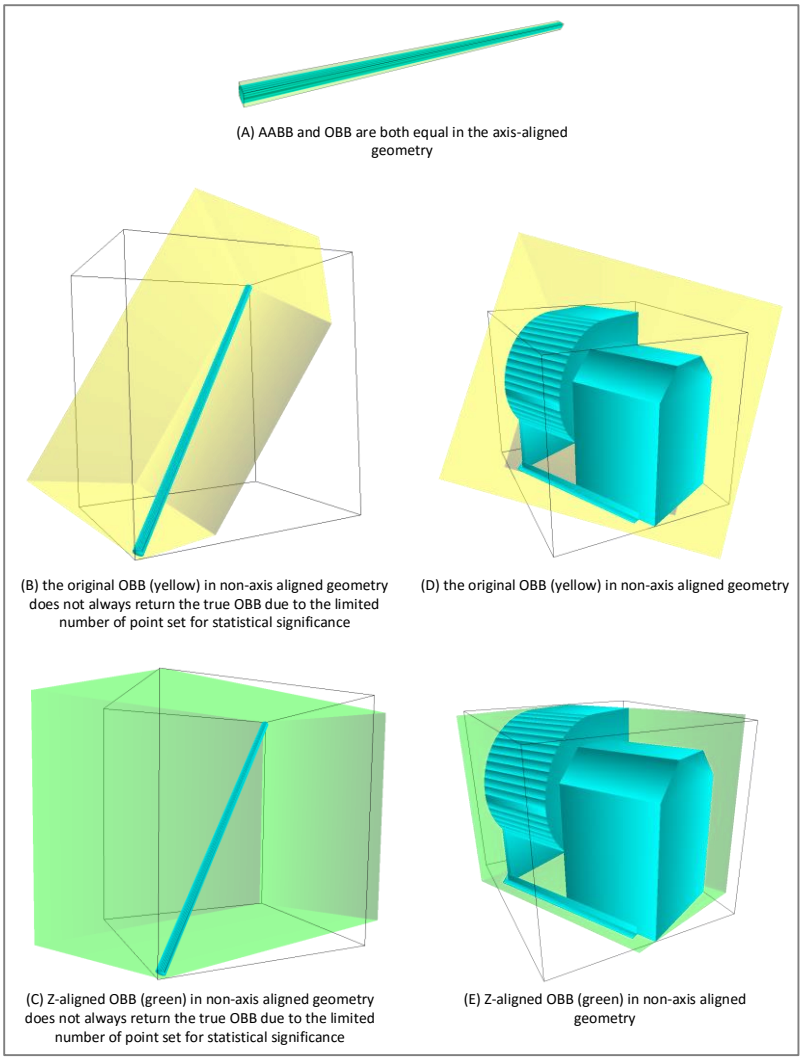


Figure 58 - AABB and OBB of a Geometry

6.2.4 Spatial index

Because geometry-based operations are computationally expensive, a fast indexing scheme is required to perform quick assessment for geometry related operations such as whether two or more objects interact in the 3D space. Oracle SDO provides spatial indexing capabilities using an R-tree index. It is an indexing scheme based on the bounding rectangle of a set of objects. This indexing scheme was first introduced by Guttman (Figure 59) [128]. The illustration shows how R-tree works in 2D. The same principle is extended to 3D by using a cuboid instead of a rectangle.

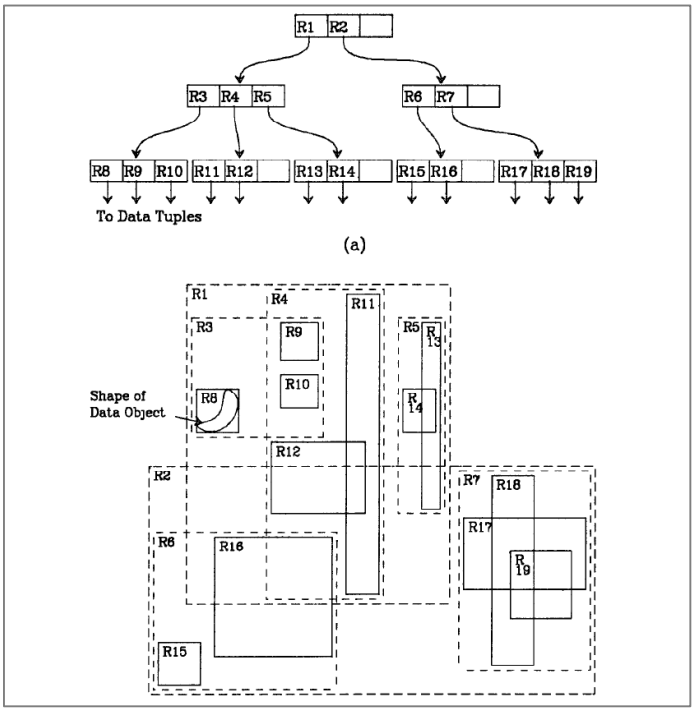


Figure 59 - R-tree Indexes

R-tree is probably the most popular indexing scheme used in the spatial world because it is compact and fast. Since the index is based on the AABB of an object and also based on a collection of objects in overlapping rectangles, it is very efficient in reducing the candidate set to evaluate. It needs the second step of the process to discover whether objects sharing the index actually interact. This second step requires the use of the actual polyhedra. In the case that data are relatively dense and may have a lot of overlaps such as typical to BIM data, the performance of an R-tree index that is built into SDO is not satisfactory for the needs of rule checking. For example, a search for a collection of more than a hundred chairs and tables in a canteen space may take around 70 seconds to complete. This is for a single space. Extending it to the entire building, which may contain more than one thousand spaces may take significant amount of time to complete.

The more suitable representation for use in BIM is an approximation of the geometry that also serves as the spatial index. This method is known as a space or cell

decomposition. There are several techniques known for spatial indexing in the 3D world. Gaede summarized several techniques and research in this area that are still relevant and used today [129]. A more recent textbook on this topic provides an excellent coverage of various techniques [130]. In this research, Octree is used due to its suitability for approximating the shape and its suitability to be encoded into one dimensional data that can be indexed with a traditional B-tree in the database, known as Z-order or Morton code, named after Morton who proposed it in 1966 [131]. The next section is dedicated to explain more details about the use of this technique in the research.

6.3 Spatial Indexing

Spatial indexing entails an approximation of the geometry in order to achieve a high performance search of relevant object inside the 3D world. Octree decomposition is a popular technique used in Computer graphics and games. Three sub-topics will be described in more detail in this section, i.e. the concept and approach for the Octree decomposition, the encoding technique for fast spatial query and a new concept used in this research, i.e. non-overlapping Octree decomposition that is suitable for a situation where many objects are overlapping as in the typical BIM data.

6.3.1 Octree indexing

Octree is a tree structure that decomposes a cell in a 3D world into exactly 8 sub-division called octants. Each of the octant in turn can be sub-divided further into 8 sub-divisions and so on until the desired level of subdivision is reached. The technique was first introduced by Meagher [132, 133]. It is an extension of a Quadtree that works for a 2D subdivision, where one region is sub-divided into exactly 4 sub-regions. In the course of explanation of the use of Octree, sometimes Quadtree illustrations are used for clarity since the concept is identical (Figure 60).

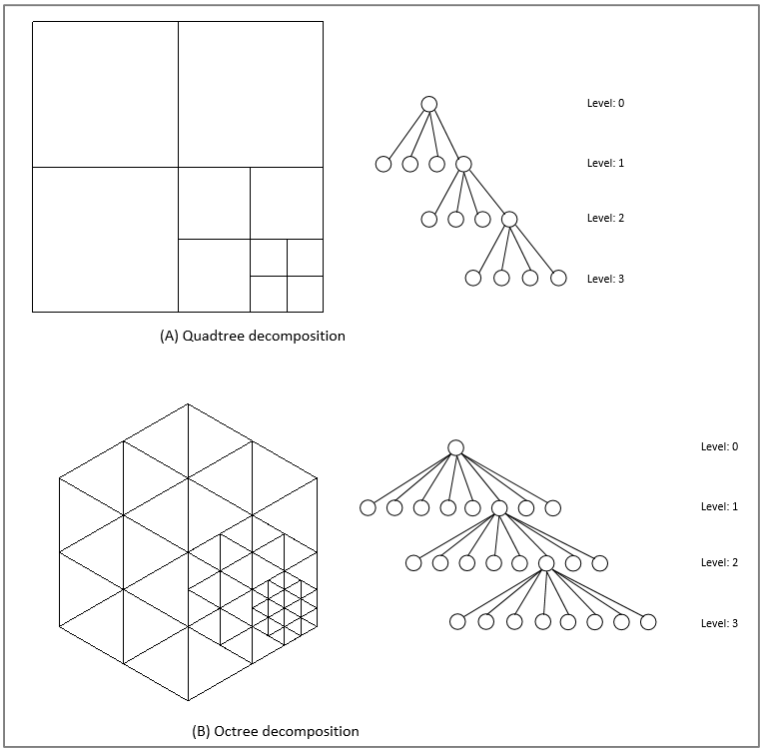


Figure 60 - Quadtree and Octree Decomposition

Using the Octree, any shape of geometry can be approximated by a set of octants that may be of varying levels (Figure 61). In this research, the following algorithm is used to perform the Octree subdivision.

Start an Octree at the root level, i.e. level 0
Perform check of a solid in respect of the Octree cell (octant)
There are 4 conditions that needs to be checked:

- Cell is disjoint with the solid; abandon the cell
- Cell intersects the solid; perform further subdivision (recursive)
- Cell completely encloses the solid; perform further subdivision (recursive)
- Cell is completely enclosed inside the solid; keep the cell and stop subdivision

This process is performed recursively until one of the cell is completely inside the solid, or the subdivision has reached the maximum level. Any octant that exist with all eight of its siblings (they may be of an intersected or completely inside type) should be merged and only the lowest level of the subdivision is kept.

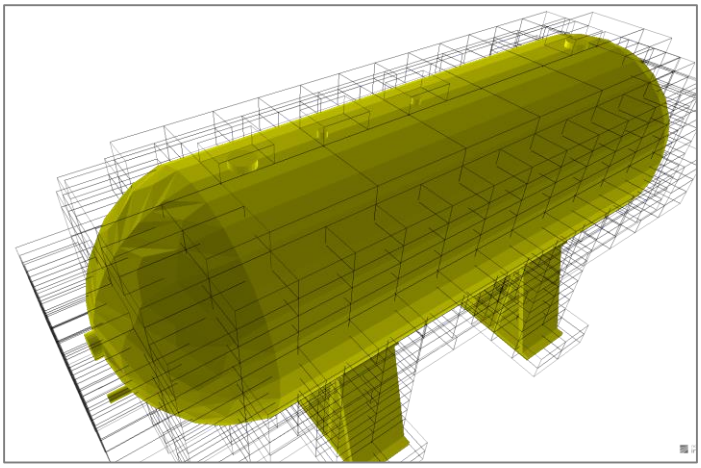


Figure 61 - An Example of Varying Cells of Octree Encoding

Two pieces of information have to be pre-determined for the algorithm to work, i.e. the world bounding cell (or the root) and the maximum level of subdivision. The more levels of subdivision used, the more accurate the approximation of the shape is going to be, but there will be a much larger number of cells too (every additional level will increase the number of cells by 8^2 times). It becomes an issue of familiar diminishing returns where further increase of the level of subdivision does not improve much of the benefit and instead increases the overhead of the number of cells for storage and cost of access. Since this is an application for building models that always corresponds to the physical reality and fixed scale relative to human and devices in buildings, an absolute size of the smallest octant may determine the suitable level of subdivision. Through intuition and many tests with various building models, a size around 200 mm or ¾’ seems to reasonably provide an optimum subdivision. Based on this smallest cell value (on the longest side of the cell), the level of subdivision can be automatically calculated using the bounding box of the model.

To facilitate efficient spatial indexing, the Octree cells are stored in the database so that the concept of “compute once read many” is maintained. This is contrary to the approach Borrmann et al. took with the Octree decomposition approach, where the subdivision is computed runtime [36, 37]. The runtime approach has severe limitations in

terms of performance and it will degrade significantly with the increase of the number of objects that need to be evaluated in pairs consistent with computation complexity $O(n^2)$. To make Octree decomposition beneficial to rule checking, a combination of strategies for encoding, database storage, and B-tree indexing, are needed to reduce the computational complexity to $O(\log n)$.

6.3.2 Octree encoding

The aim to store the octree cells are twofold: to store the outcome of the computation so that it is done only once, and to provide efficient access to the indexes for any subsequent queries. To store the Octree cell into a database, the encoding method known as Z-order or Morton code is used [131]. It basically sets the ordering method following the axis in a Z like order. In this case, a binary code is used to identify the cell and they are organized following the Z-order (Figure 62(A)). To encode one cell uniquely within one level of subdivision, a 3 bit code is needed to represent all the 8 octants. They are then packed by sequencing them into an array of 3-bit code (Figure 62).

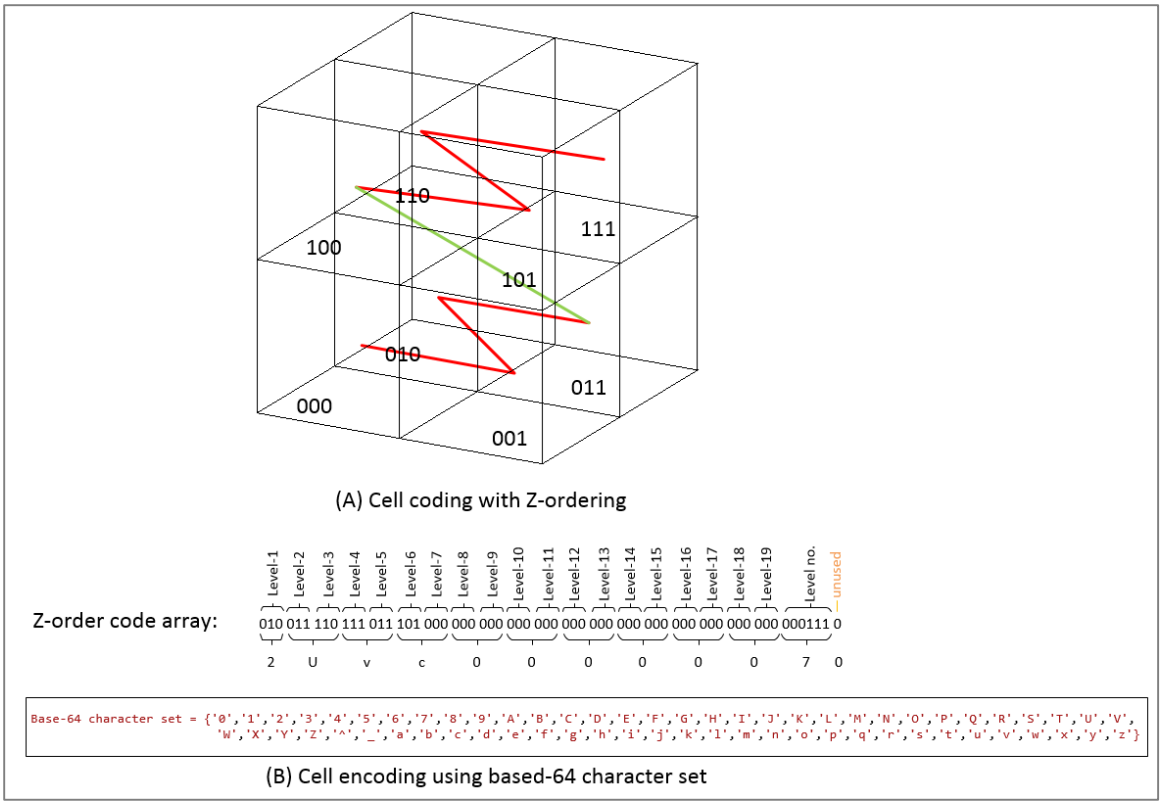


Figure 62 - Octree Cell Coding Using Z-order and Character Encoding

A 64-bit number is used to capture the code. There are 19 levels of maximum subdivisions that requires 57 bits, leaving 7 bits remaining to keep the level number of the cell. To allow for a more intuitive use of the index by allowing the traversal of the tree to be done without the real need to convert into the code, a base-64 character encoding is used that represents each of 6 bit codes (Figure 62(B)). The list of characters used are selected to exclude unreadable characters, but still maintain the sequence of ASCII/UTF code. With this encoding, the cell ids can be stored in a database as a simple character column and this column can be indexed with just a standard database B-tree indexing. For example, for an octree cell of code 2Uvc00000070 in a 3D world coordinates of [(-141.70335, -281.48334, -.16491666), (413.484192, 350.934753, 313.262238)], is a cell at level 7 with a Z-code 4F7740000000000E (in Hex) or 010 011 110 111 011 101 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000 111 (in the binary sequence format). Its index location (based on lower left bottom location of the

Octree cell) in the world coordinates is at [(57.81717290625, 331.17168759375, 63.4999741303125)]. To find all cells that are the descendants (traverse down) of this cell a database query can be done by simple use like condition: (CELLID LIKE '2Uvc%' AND CELLID > '2Uvc00000070'). Or to find the parent(s) of this cell:

- Immediate parent: 2Uvc00000060
- All ancestors:
  - At level 6: 2Uvc00000060
  - At level 5: 2Uv000000050
  - At level 4: 2Us000000040
  - At level 3: 2U0000000030
  - At level 2: 2O0000000020
  - At level 1: 200000000010
  - At level 0: 000000000000

Finding an enclosing Octree cell for a point p in the 3D world at a specific level of subdivision b is also straightforward by computing:

Axis X: cell code at X = floor(p.X - WorldBB.LLB.X /cell_size_at_level_b_at_X)

Axis Y: cell code at Y = floor(p.Y - WorldBB.LLB.Y /cell_size_at_level_b_at_Y)

Axis Z: cell code at Z = floor(p.Z - WorldBB.LLB.Z /cell_size_at_level_b_at_Z)

The Octree cells and encoding are done for individual objects. Each object will have a set of Octree cell indexes stored in the database. Since the cells are derived from the same world space, they are capable of answering questions related to the binary topological relationships known as the 9-Intersection Model used widely in the GIS domain [134, 135]. The 9-Intersection model is derived from a matrix of relationships between two arbitrary elements A and B. For each of the elements in the 2D world (It is extensible to 3D too), it has three distinct subsets $\{A^o, \partial A, A^-\}$, $\{B^o, \partial B, B^-\}$, where A^o is the interior set of element A, ∂A is the boundary of element A, and A^- is the complement

set of A. The 9-Intersection is defined using the matrix intersection between elements A and B as follows:

$$I = \begin{pmatrix} A^o \cap B^o & A^o \cap \partial B & A^o \cap B^- \\ \partial A \cap B^o & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^o & A^- \cap \partial B & A^- \cap B^- \end{pmatrix}$$

Figure 63 shows the matrix for the 9-Intersection Model with the matrix I of the values of the matrix intersection above using an empty set ($\emptyset$) or a non-empty set ($\neg\emptyset$). The Octree cell subdivision offers quick answers to the questions of disjoint, contains, inside, equal, and overlap using its approximate shape represented by the Octree cells. In this case, covers and coveredBy are mostly identical to contains and inside respectively. The only relation that the Octree decomposition alone is not sufficient to answer is the meet relation. It can only be answered in combination with the boundary faces that is described in 6.2.2 and its boundary relationship. In many cases of BIM rules, the points of interest to check are that there is some form of intersections, overlap, contains, inside, covers, coveredBy and equal. The Octree cells are sufficient to satisfy those cases.

Additionally, spatial relations often require non-intersection relationships to be queried. For example, queries related to a relative distance such as above, below, left of, right of, etc. Frank introduced a concept for cone-shaped and projection-based models of relationships between points [136], and Goyal introduced a concept called Cardinal Direction Calculus (CDC) for representing relational operators between two 2D regions [137]. In a 3D model, the spatial relational operator becomes harder to compute unless they fit simple conditions that can be fulfilled by both approaches by Frank and Goyal. Borrmann et al. improves upon the work by introducing strict and relaxed modes in a halfspace-based model [109]. The difficulty in finding the exact relation in the 3D space is illustrated in Figure 64. In those examples there are cases in which a straightforward definition of spatial relations are not valid. In these cases, each of the rules that may need such a relationship may need to define the local context of the relations depending on the

semantics. In general use, the use of a very simple bounding box with 6 relations (above, below, leftOf, rightOf, front, behind) can be used for a fast assessment. The more specific relations that may depend on the semantics should be dealt with in a specific context. For example an ADA rule that checks the danger of overhead obstruction by a staircase will require specific consideration of what is below (Figure 64).

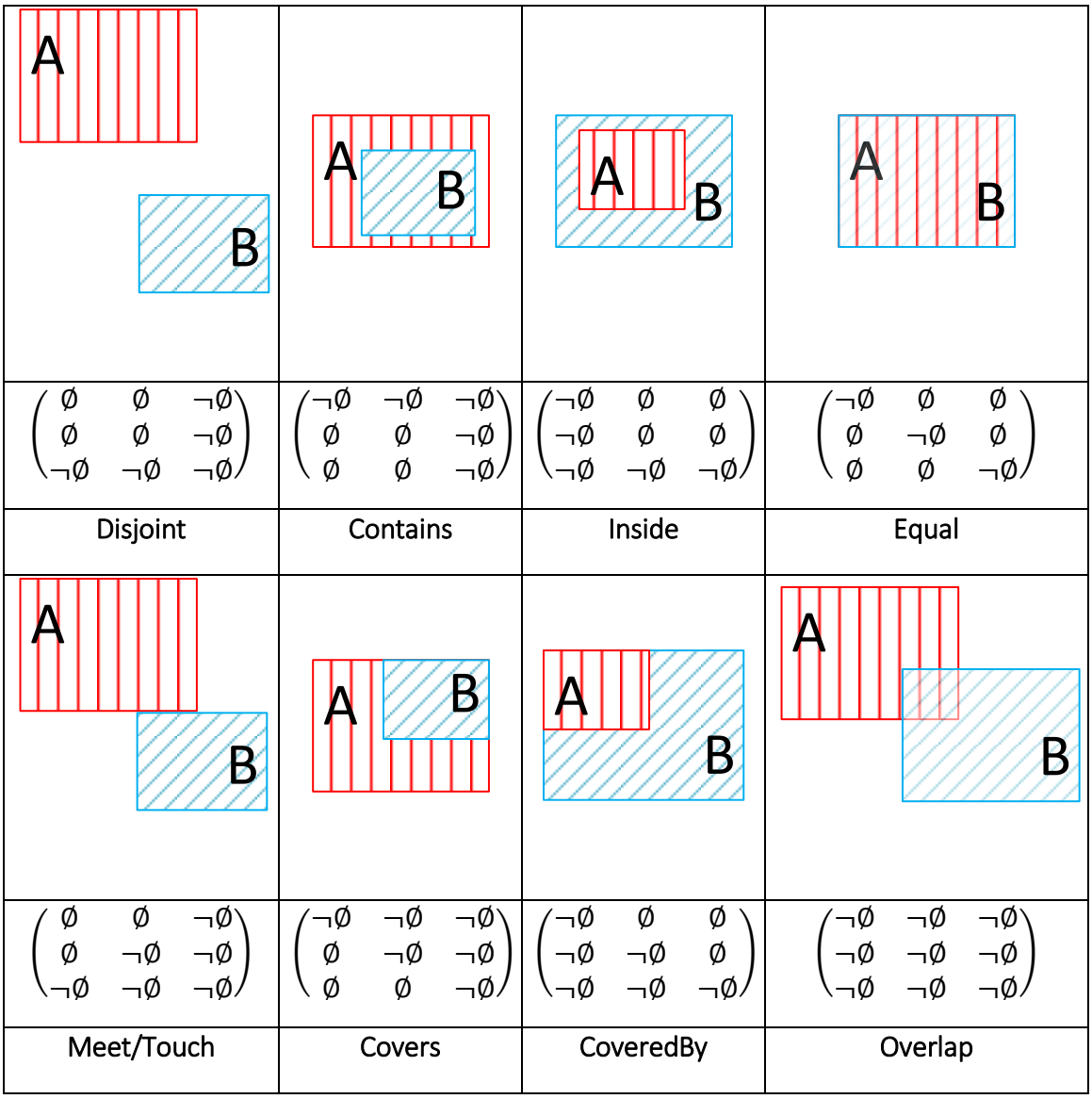


Figure 63 - Examples of the Topological Relations between Two Spatial Regions in 2D

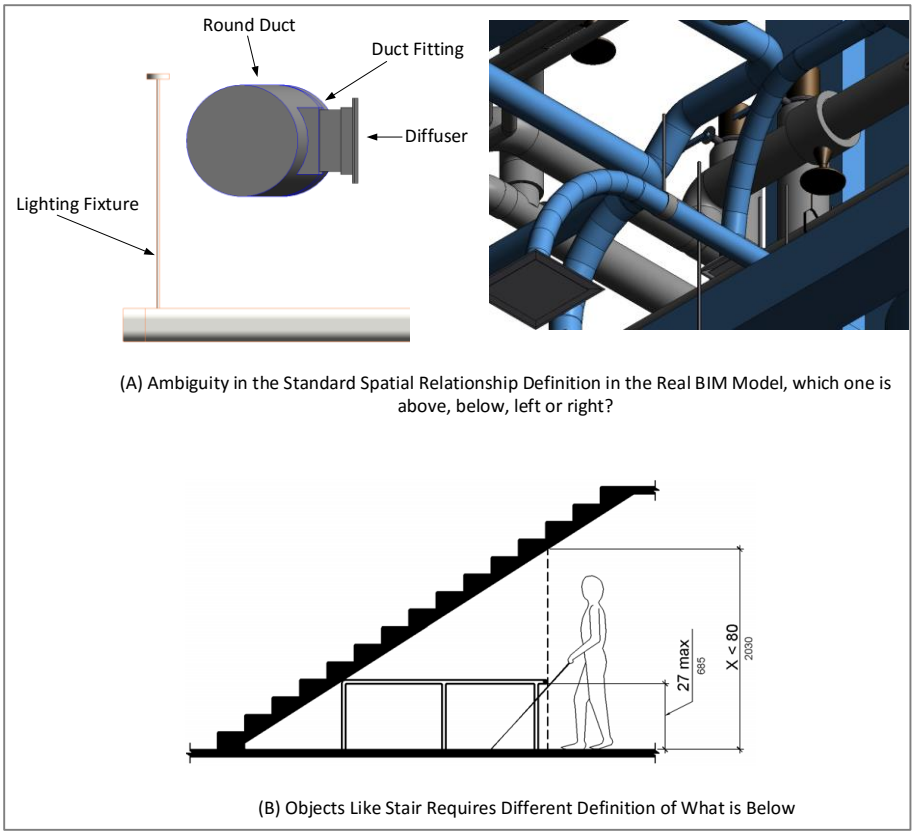


Figure 64 - Possible Ambiguity in the Determination of the Spatial Relationship in 3D

In this research, the Octree cell codes can be used to support the 6 relations based on the bounding box of the Octree decomposition, with 6 additional relations (directly above, directly below, directly leftOf, directly rightOf, directly front, directly behind) that are based on the constraints of the bounding box of the object of interest (Figure 65).

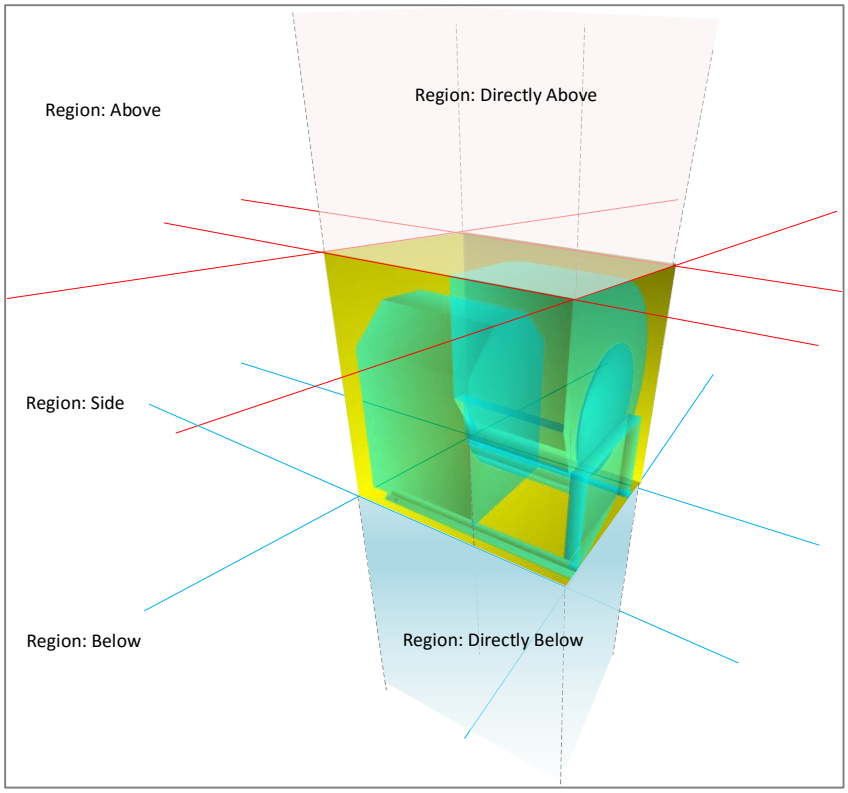


Figure 65 - Regions that Can Be Determined Using Octree Bounding Box

6.3.3 Non-overlapping Octree indexes

Building objects are not uniform and vary widely in terms of size, shape, position and orientation. Furthermore, one feature that BIM data has is that many objects are expected to overlap (e.g. between a space and all the objects inside the space and all other objects that may pass through the space such as MEP services). The MEP services also often will penetrate (intersect) the structural and architectural objects horizontally and vertically. The Octree subdivision for each of the objects may not be exactly the same since they are of different sizes. Using the technique to traverse up and down the tree to find the same cell ids between two or more objects are possible, however the query becomes more complex and harder to manage when there are more objects involved. For example, to identify the headroom clearance of a space, the space geometry must be subtracted with any object that is inside or that protrudes into the space. The remaining space that is not subtracted will determine the headroom clearance. To do this, the spatial

operators inside, overlap, and coveredBy have to be evaluated against all objects that may be present within the space. The object may vary from a relatively large duct to a really small object like a sprinkler head. They may involve hundreds or even thousands of such objects and therefore to perform traversal of the Octree becomes impractical. To overcome that a new approach is introduced to create non-overlapping Octree indexes.

The basic principle of non-overlapping indexes is that all the leaf nodes of the Octree in the final form should not contain any cells that have an ancestor – descendant relationship. To achieve that, during creation of an Octree for a particular object, a "master" Octree needs to be created that keeps the global Octree cells and maintain the catalog of objects that own each of the leaf nodes. Whenever another object that shares the same tree branch is indexed, three possibilities need to be evaluated:

(i) Is the same leaf node already present in the master tree?

(ii) Does the leaf node have an ancestor in the master tree?

(iii)Does the leaf node have descendants in the master tree?

If condition (i) is satisfied, it is a straightforward case where the object will have the leaf node as one of the indexes and the master tree just needs to add the information that a new object now owns the same leaf node. For the other two cases, there is more work to do:

- An ancestor leaf node is found

  In this case, the smallest subdivision will be preserved, i.e. the new leaf node will become the leaf node in the master tree. But because of the principle of non-overlap, no ancestor leaf node should exist in the master tree. Therefore, the ancestor leaf node needs to be subdivided until it reaches the level of the new leaf node. In this process, all the objects that currently own the ancestor leaf node must be transferred to all the descendants. Figure 66 illustrates this process.

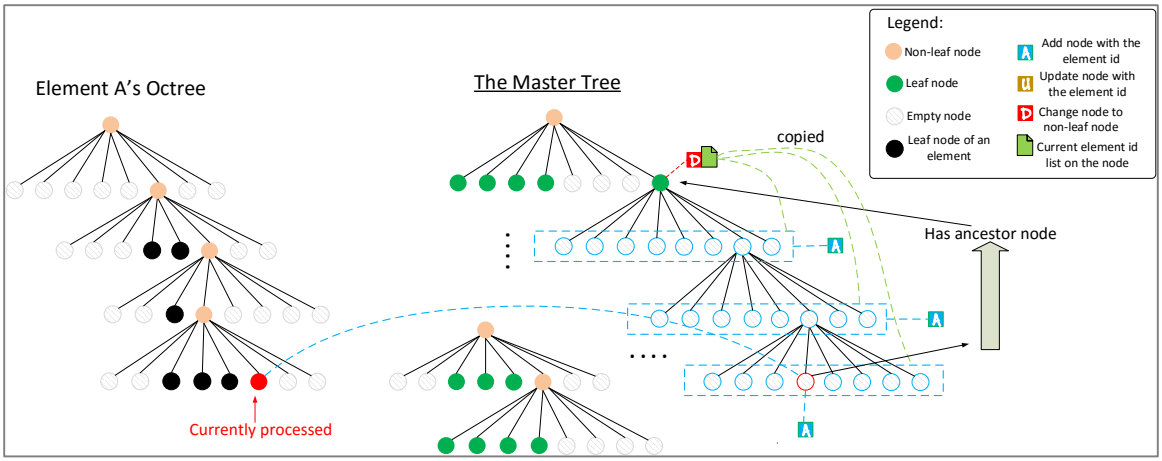


Figure 66 - Octree Insertion when an Ancestor Node Found

- Descendant leaf nodes are found

  If the descendant nodes are found, the current leaf node need to be subdivided on each branch following the existing descendant leaf nodes. It stops subdivision at the level where the descendant leaf nodes are in each of the branches. In this case, one leaf node will become a set of descendant nodes. Figure 67 illustrates this process.

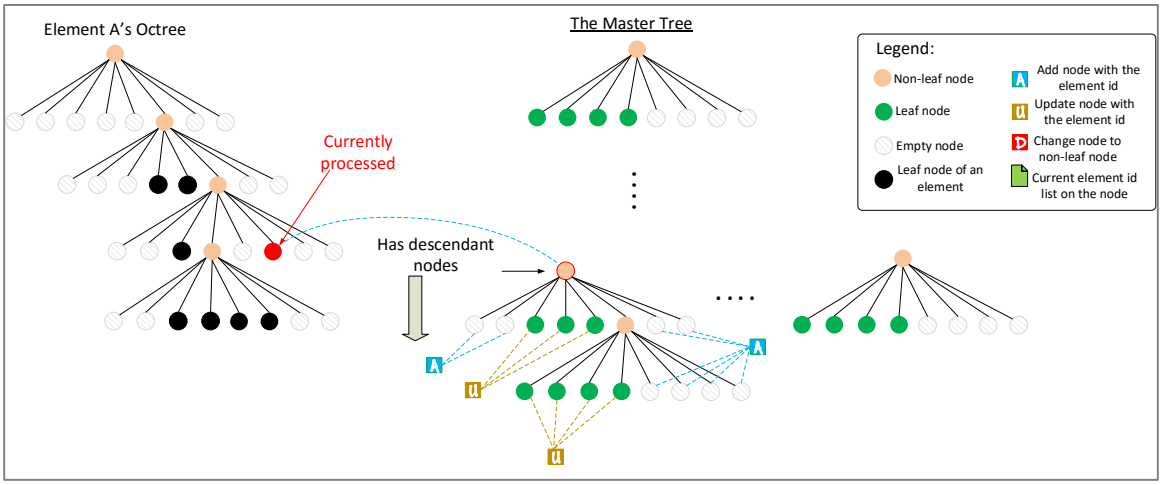


Figure 67 - Octree Insertion when Descendant Nodes Found

The final Octree cell indexes are created based on the master Octree. It will contain non-overlapping Octree cells (Figure 68). With the non-overlapping cells, query

for the 9-Intersection operations are much more simplified. Consider the following examples:

- For an object o in a 3D world Euclidean space $\mathbb{R}^3$, find all other objects that overlap o. Overlap queries in this case will include overlap, equal, inside and coveredBy. To be more specific for only overlap conditions, additional queries may be needed:

  $$Overlap\ set\ \phi = \Psi_\sigma \cap \Psi_\Upsilon$$

  where $\Psi_\sigma$is a set of Octree cells belonging to the object σ

  Ψ_Υ is a set of Octree cells belonging to the set of objects Υ

  In an equivalent SQL query statement, it looks like:

```
SQL> select a.elementid, b.elementid
   from bimrl_spatialindex a, bimrl_spatialindex b
   where a.cellid = b.cellid
         and a.elementid=' 3nHD$X6Cf5Pve_4$zXCoFO';
```

- For every space in S, where S is a collection of spaces in the 3D world Euclidean space $\mathbb{R}^3$, find all spaces that do not have one or more sprinklers installed:

  $$set\ of\ spaces\ \phi = S - S(\Psi_S \cap \Psi_\Upsilon)$$

  where $\Psi_S$is a set of Octree cells belonging to the spaces in S

  Ψ_Υ is a set of Octree cells belonging to the sprinklers in set Υ

  In an equivalent SQL query statement, it may look like

```
SQL> select elementid
         from bimrl_element
         where elementtype='IFCSPACE'
   minus
     (select unique a.elementid from bimrl_spatialindex a,
         bimrl_spatialindex b, bimrl_element c,
         bimrl_element d, bimrl_type e
       where a.cellid = b.cellid and c.elementtype='IFCSPACE'
         and a.elementid=c.elementid and
         d.elementtype='IFCFLOWTERMINAL'
         and b.elementid=d.elementid and d.type=e.elementid
         and e.ifctype='IFCSPRINKLERTYPE');
```

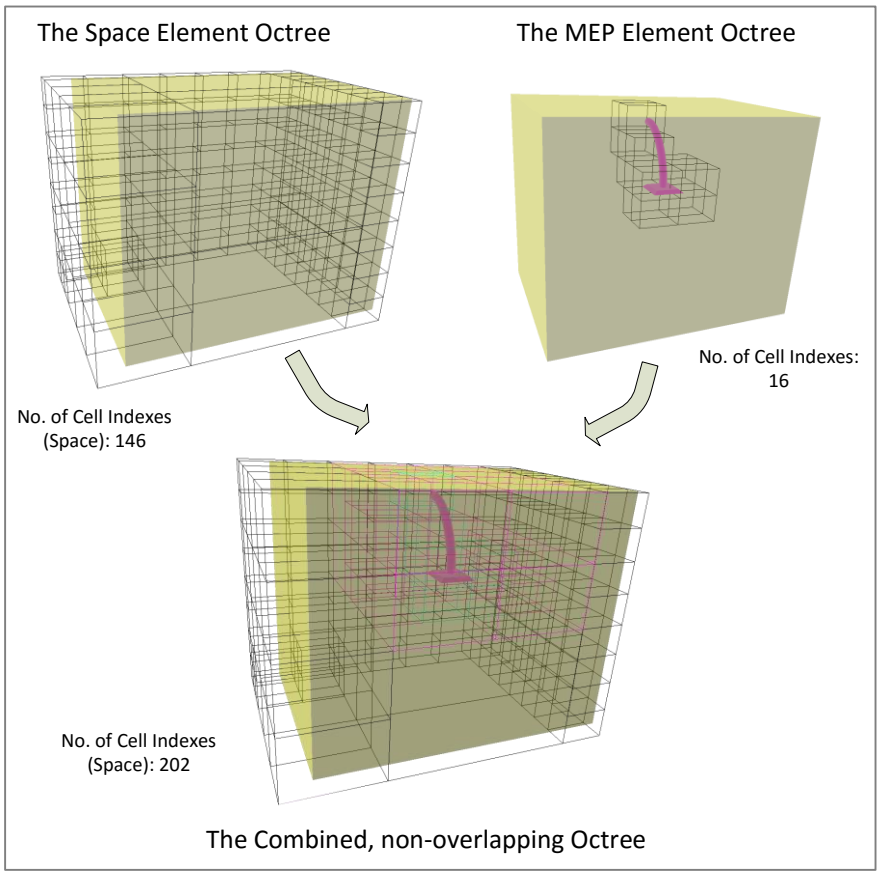


Figure 68 - An Example of the Non-Overlapping Octree Cell Indexes

6.4 Transient Geometry

In the actual BIM rule checking, many times the spatial relation check required is not the type with unlimited distance (halfspace), but often it is of a finite distance. For example in the OSHA rule for protection from falling: it is specified that a protection from falling in the construction area is needed when there is a risk of falling at more than 6 feet (1.8 meters) [78]. The spatial relation needed here is that the check for any protection below is only relevant if it is less than 6 feet (1.8 meters). Any object that may provide protection below but is more than the specified distance does not give any relevance to the rule anymore and should not be considered at all. Another example from International Fuel Gas Code (IFGC) 2009 is clause 620.4 that specifies the requirements

for clearances of combustible materials above, around and below a unit heater. The distances are 6 inches (152 mm), 18 inches (457 mm) and 12 inches (305 mm) respectively. In those checks, a transient geometry should be created and then used to perform the intersection check using the 9-Intersection models. The transient geometry is not known in advance and therefore it is not possible to be created beforehand and cached or stored in the database. It will be very useful to support such creation of the transient geometry in the runtime when a specific object and checking requirement calls for the need to create it.

In this work, this unique capability is supported as part of the integrated facility that can be invoked through the language interface (BIMRL) through CONSTRUCT statement described in the next chapter. The generation of the transient geometry usually works in tandem with the objects being evaluated. For example to check for the danger of falling in the OSHA rule mentioned above, the transient geometry that needs to be created should be generated relative to the edges of the slab where there is a danger of falling. The transient geometry must be created along the edges and is aligned to the orientation of the edges that changes as it goes along the shape of the slab (Figure 69). For this purpose, the information about the side faces of the slab will be useful for the relative position, a starting edge and the normal direction to determine the starting face of the transient geometry (topology face information from the indexed faces).

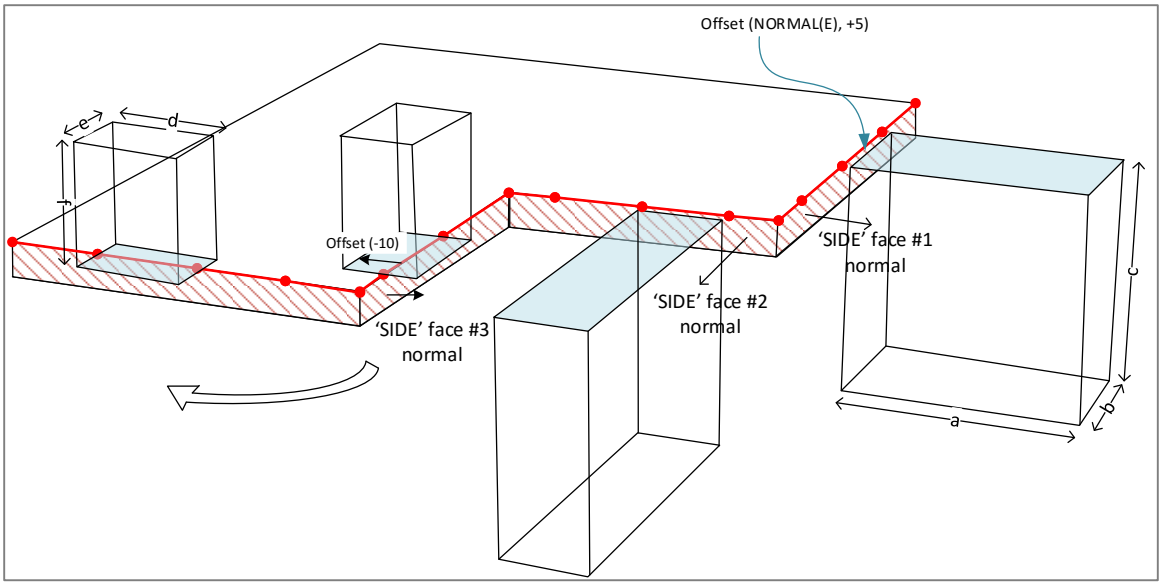


Figure 69 - Creation of the Transient Geometry using Indexed Face Information

There are a few simple geometry types supported currently. They are line and linestring, 3D planar face, simple AABB, simple (straight) extrusion that may involve different start and end faces but they must share the same number of vertices and sequence, and Brep geometry from the list of faces. Internally the transient geometry is stored in a temporary table using the same representation as the building model using Oracle SDO. The geometry may also be indexed so that it may participate in the spatial operations described above.

6.5 Selecting a Suitable Form of the Multiple Representation in Rules

The multiple representation concept provides options that a rule definition can use to perform the most suitable form of representation that is sufficient to solve the problem at hand. The use of different levels of details from the multiple representations will not create any inconsistency of the final outcome, except that it affects the granularity of the outcome due to the approximation that is used in different representations. The best method that should be adopted in BIMRL rule definition is to use the "least is best"

approach, i.e. to choose the least precise representation that provides a good enough answer to the problem to be solved. Selecting higher precision information will give only a marginal improvement in terms of the result while requiring much more significant effort to write the rule and increasing the runtime cost.

CHAPTER 7

BIM RULE LANGUAGE (BIMRL)

In Chapters 5 and 6, BIM data is turned into a query-able database complete with support for geometry and spatial queries. The third component for an efficient rule checking system is a standardized rule language. The language needs to combine the flexibility of the query-able database and the need to support specific structure of rule checking as described using CG (CHAPTER 4). It seems reasonable to extend an already established SQL language for rule checking purposes because the data is already available inside a database, and as discussed in CHAPTER 4, the query component is an essential part of a rule language. Beside the query capability, BIM rules typically consist of the following critical components:

- Projection, which includes the basic building elements and also derived objects, properties and dynamically constructed geometry
- Query capability. The query capability that includes filter, joins and aggregate capabilities
- Evaluator. The evaluator requires arithmetic operators, set operators such as IN and JOIN, spatial operators for geometric and spatial based queries and graph based queries.
- Action. It defines what action that needs to be performed responding to the results of the Evaluator. Action can be a terminal action such as printing the message, or it can also be a transient one such as keeping the evaluator result in a table, or if the data model allows by updating the object with appropriate information from the evaluator.

The use of language for the purpose of solving BIM rule checking is not entirely new. There are several precedents that have been done. They provide inspiration for

finding a suitable language specific for BIM rules. The previous works also provide clues of what works well and what limitations are present.

7.1 Previous related works

Four relevant areas of previous works are included in this review. They are: query languages, query languages supporting spatial operations, rule languages and combined query and scripting environment.

7.1.1 Query languages:

- SQL (Structured Query Language)

  SQL is an established query and manipulation language used to manage data in a relational database management system (RDBMS). It was based on a paper by Codd from the work in IBM that is based on an algebra of relations [138]. It has become ISO standard since 1987, and its latest version is SQL:2011. SQL provides flexible query capability that can deal with a simple query to a very complex queries that may be formed by sub-queries. The concept of everything is a relation is a powerful concept in relational database. Everything is represented as a relation including the query result. Therefore a very complex queries can be achieved by simply combining many sub-queries. The SQL language is a mature specifications and it has been extended to include spatial data and query, and object relational feature. There have been previous efforts to build IFC model server using XML and RDBMS [97], and using Object-Relational Database IFC server [99]. It was also proposed for the use in the downstream model handover scenario by de-normalization of the IFC into tabular tree based connections [139]. So far, the use of relational database for IFC has not been very successful due to its performance issue and complexity of building SQL queries for IFC that is very hierarchical and with a lot of nested relationship. Mapping each IFC entity to a

relational tables will create more than 600 tables for IFC2x3, which make even a simple query to the database for an IFC entity may require many table join statements. Despite the challenge, the use of RDBMS as the BIM database still has its appeal due to established infrastructure support around it and extensive use in the enterprises. The later reason is important since often BIM is not standalone but increasingly needed to be integrated to the rest of the enterprise database which is mainly RDBMS.

- SPARQL

  SPARQL is a query and manipulation language for RDF (Resource Description Framework) graphs, which are used in semantic webs [140]. SPARQL uses SQL-like language construct that includes SELECT statements and similar functions such as aggregation functionality: GROUP BY, ORDER BY, DISTINCT, and other query modifiers such as CONCAT, EXISTS, HAVING, etc. The main difference is that SPARQL does not deal with tables but with Resource as the object. SPARQL is based on graph pattern matching operating on RDF graphs that is basically a three-tuple of Subject – Predicate – Object, where Subject and Object are represented as the graph nodes and the Predicate as the edge. A small distinctive style of syntax used in SPARQL is a variable name that starts with '?' or '$', e.g. ?varname, and also ability to create blank node for later assignment using ':' prefix, e.g. :predicate1. BIMQL, which will be described later in more detail, uses some of these styles in the language definition [42]. In the implementation of SPARQL and RDF, many have suggested the use of RDBMS as the database for the RDF graph, making it facing similar challenges with IFC in RDBMS [141]. SPARQL also supports queries from federated sources.

  Pauwels et al used SPARQL for rule checking for building performance [14], and Beetz et al used SPARQL for checking IFC models [142].

- PMQL

  Adachi developed Partial Model Query Language (PMQL) as a query language to access partial models from the IFC model server [143]. It describes the partial model object structures by XML element structures. It introduces `<cascade>` to deal with recursive traversal of IFC object structure. The data returns from PMQL query are in form of BLIS-XML and STEP P21 file formats.

- BIMQL

  BIMQL is an effort to provide a standardized query language with SQL-like syntax and intended to support not just queries but also the manipulation capability known as CRUD (Create, Update, Delete) [42]. The language specification allows queries to recursively traverse the IFC object structure following the idea started with PMQL. It is implemented on top of BIMserver, but it is currently only on a prototype level.

7.1.2 Languages Supporting Spatial Query

OGC (Open Geospatial Consortium) defines a spatial extension to SQL for its GIS simple feature specifications. The standard includes support for geometry storage and appropriate spatial query operators based on the 9-Intersection model: equals, disjoints, touches, within, overlaps, intersects, contains, etc. [111]. It presently supports only 2D geometries that are typical for GIS data. With the increasing need to support not only a flat 2D map, e.g. a city map, support for 3D geometry has started to make its way into the implementation in various RDBMS products. Among them are Oracle Spatial [114] and PostgreSQL with PostGIS extension [113].

Borrmann et al. proposed a spatial based query language to support BIM queries using an octree approach, which can be implemented as an extension to SQL using the object relational feature in SQL:1999 standard, function in pre object relational SQL-92, or embedding the spatial operators in XQuery [35]. Further update to this approach uses a

combination of Brep and R-tree indexing [40]. This approach still has its limitations in terms of performance since it is based on pair based evaluation. As predicted in 6.3.1, computational complexity for such an approach will take the form of $O(n^2)$. The use of R-tree indexing reduces the computational complexity to $O(n)$ as reported in [40].

7.1.3 Rule languages:

- NRL (Natural Rule Language)

  The NRL is a language that allows users to constrain, modify and map data in a diverse format. It is designed for automatic translation to execution languages [144]. It has a two part specification: constraint language and action language. The constraint language is a language for expressing rules that constrain data models, i.e. to provide static semantics for models. The action language is an extension to the NRL that allows rule writer to specify certain actions, e.g. creation or deletion of objects and setting values that should take place when the conditions as specified in the rule hold.

  The NRL aims to provide a user readable syntax friendly to non-developers, which uses an English language style alternative to FOL (first order logic). The general form of the language looks like:

  If <constraint> then <action> [else <action>]

  For example:

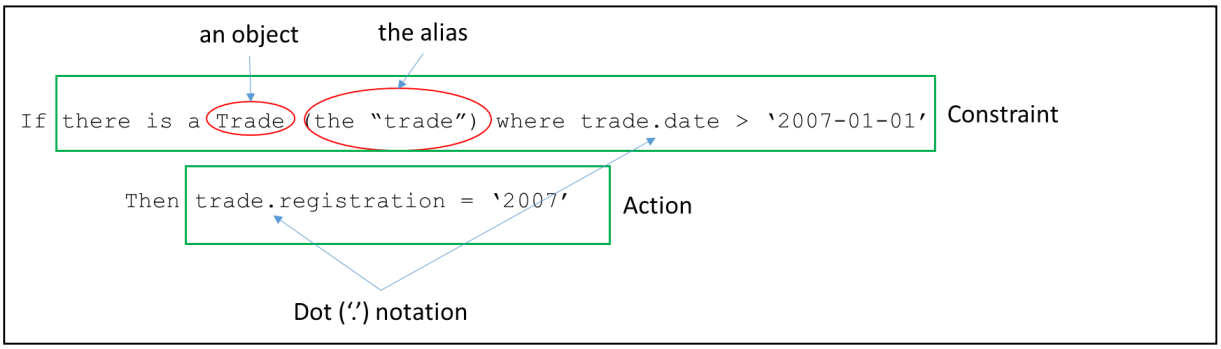


Figure 70 - An example of NRL sentence

- SWRL (Semantic Web Rule Language)

  SWRL is a standard proposed by W3C. It is a rule language combining OWL (Web Ontology Language): OWL DL and OWL LITE, combined with the Unary/Binary Datalog RuleML sublanguages of the RuleML (Rule Markup Language). The language takes the form of an implication between 2 main elements, an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold [145].

  For example: a rule that asserts that the combination of the `hasParent` and `hasBrother` properties implies the `hasUncle` property.

  - In the informal form:

  ```
  hasParent(?x1,?x2) ∧ hasBrother(?x2,?x3) ⇒ hasUncle(?x1,?x3)
  ```

  - In the abstract form:

  ```
  Implies(Antecedent(hasParent(I-variable(x1) I-variable(x2))
            hasBrother(I-variable(x2) I-variable(x3)))
        Consequent(hasUncle(I-variable(x1) I-variable(x3))))
  ```

  An example for effort for rule checking using SWRL is done by Zhong et al for construction quality control [62].

- LegalRuleML

  LegalRuleML is an XML format for capturing legal text into the formal descriptions of norms, guidelines and legal reasoning [146, 147]. It is a specialization of RuleML of the legal domain that carries a specific semantic. It also aims to capture different legal interpretation and the changes over time. The potential use of the language to capture building codes was proposed in [5]. As the effort is still in its early stages, it remains to be seen how much it may help in the code checking automation effort.

7.1.4 Query and scripting/programming environment:

- EQL (Express Query Language) & Tnr (Templates and rules)

  EQL is an SQL-like query language for to perform query on P21 file. EQL allows ad-hoc and partial model access to P21 file with similar power and flexibility of SQL [148]. EQL itself is not sufficient for use in rule checking. Tnr was introduced to complement EQL as a scripting environment making use of EQL and other types of access to different sources [149].

- BERA (Building Environment Rule and Analysis)

  BERA is intended to be the rule language for BIM [1]. It introduced a very similar approach as EQL and Tnr, i.e. a query language and a scripting environment. BERA also introduces the concept of BOM (Building Object Model), a simplified version of BIM, and is intended to be a neutral representation not specific to any BIM format. BERA is not yet complete as a language. The scripting language supports only rudimentary queries to the BOM. While it is intended to be a platform that allows extension, BERA language specification lacks of depth and it requires modification to the syntax to extend the language.

- LINQ

  LINQ is an interesting addition to C#.NET programming language. It introduces a query language right into the C# programming language. The format for LINQ is similar to that of the SQL query. LINQ however is not SQL. It is intended to provide a simple and standard way to query data from various sources including SQL. One can write a provider to allow a hook into LINQ and provide access to their data from a LINQ query. The potential of LINQ for rule checking has been explored in [23].

7.2 The BIM rule language structure

From the review of previous work in the earlier section, it is clear that the expressive power and flexibility of the SQL style query language makes it a good candidate to support BIMRL. Other query languages like SPARQL and LINQ provide testament to this and also gives ideas for using similar styles especially when dealing with objects and attributes using dot ('.') notation. BIMRL will have the flexible query capability not only to filter selection but also to include various set operations including IN and JOIN. Since the query alone is not sufficient for building rules, supports for evaluator and action that are sub languages by themselves with capabilities to support certain "query"-like expressions are introduced. One critical and inseparable requirement for building rule checking is support for geometry or spatial and graph based queries. BIMRL adopts a spatial query extension similar to the OGC spatial query extension to SQL for its spatial support [111]. This choice is both practical and strategic. Practical since we do not have to reinvent specifications already defined and used. Strategic because potentially BIMRL can make use of the available technology from Oracle Spatial and PostGIS/PostgreSQL that implements this capability and extends it to support 3D geometry.

7.2.1 Key concepts

The following descriptions highlight several important concepts of BIMRL:

7.2.1.1 Relational algebra.

It uses a relational algebra supported by the underlying RDBMS. This means that the major part of the BIMRL statement is eventually translated into the SQL statement to fetch and analyze the data. It also means that BIMRL utilizes the flexibility offered by the

relational database and its features such as support for geometry in the database and various optimization methods.

7.2.1.2 Geometry construction

One important support required in a more complex classes of rules is the ability to construct a transient geometry for spatial based query against the building objects. Ability to define a geometry and place it at a desired location is critical to provide a rich set of checking functionalities of building rules. BIMRL supports geometry construction right into its query language. The geometry types supported includes line, face, bounding box, sweep and Brep (Chapter 6.4).

7.2.1.3 Multiple sets

Many of the BIM rules require the interaction of distinct sets that need to be evaluated, for example in a patient room visibility rule, distinct sets may include sets of Nurse Stations, sets of Patient Rooms, and sets of any other elements that may block the visibility. To support multiple sets effectively, BIMRL supports multiple subqueries resulting in the user named sets for later use in the evaluation.

7.2.1.4 Evaluation functions and extensibility feature

BIMRL provides some general functions that are often needed in BIM rules, such as intersection function. These functions can be used to define rules. Recognizing that there is no system that can cover every possible rule definition, BIMRL allows extensions. This can be achieved by using the language syntax intact but introducing the specific logic for checking into the evaluation function. The extensibility can be implemented with a standard popular plug-in mechanism that allows new functions to be added into the language over time without changing the language.

7.2.1.5 Chained Rules

Many rules, especially those found in the building codes, are a form of combination of many smaller rules linked together by complex conditions. In CHAPTER 4, a systematic way to break down the rules into atomic rules have been discussed. In applying those rules, often one rule is dependent on a result from another rule. To allow such a complex relationship between rules in BIMRL syntax, chaining several evaluation functions may be necessary. BIMRL makes chaining possible by the principle of relation algebra, i.e. the result one rule or statement is fed into another statement or rule. There are two ways this is possible in BIMRL:

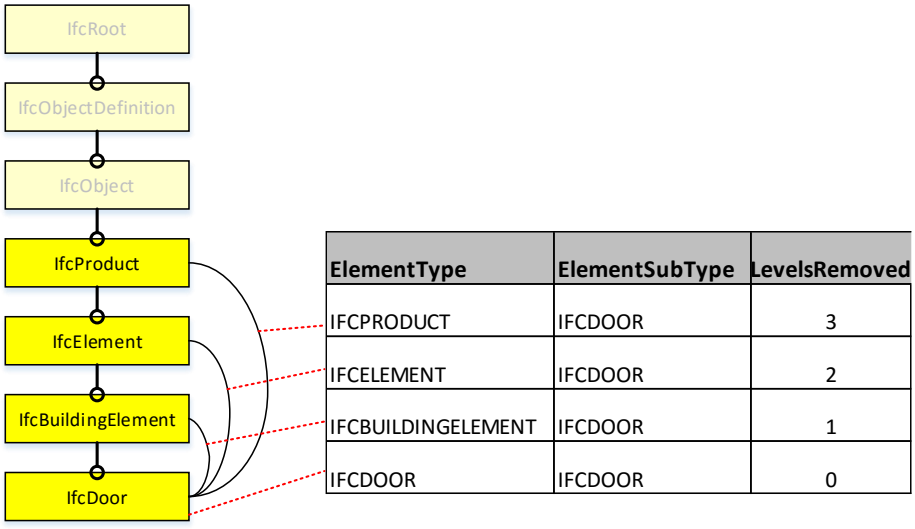
- Using chained evaluation functions within one rule
- Using multiple rules

In both cases, the result from the previous function or rule is stored in the database relation (table). The difference is in the lifetime of the intermediate table. If the chained rule works only within the rule, i.e. integrated closely with the CHECK and the previous evaluation function, it is a candidate for using a chained evaluation rule where the result of one evaluation function is fed into the subsequent evaluation function(s). The lifetime of this intermediate table is only within the rule and not beyond it. When such an intermediate result is needed by more than one rule or it is needed by a rule that has a very different set of criteria, the use of a permanently stored database table from one rule should be used. It is achieved by defining more than one rule, executing the rules in order and each rule saves the result into a table, which can be used in the subsequent rules. In theory there is no limit to the length of the chain, but making a large or long chain may

create unmanageable statements. This is undesirable as it is usually difficult to identify where it may have gone wrong in the event of an inconsistent result.

7.2.1.6 IFC Object Hierarchy

IFC objects are defined in the object oriented style. Each of the object is made of a series of inheritance (hierarchical) structures. Due to the de-normalization of the data, the hierarchical structure is "lost" and therefore there is no direct way to retrieve all "IfcDistributionElement" as an example. A data dictionary BIMRL_OBJECTHIERARCHY is defined that stores the IFC object hierarchy in flattened style beginning from IfcProduct. Figure 71 shows an example of object hierarchy information for an IfcDoor.



ElementType	ElementSubType	LevelsRemoved
IFCPRODUCT	IFCDOOR	3
IFCELEMENT	IFCDOOR	2
IFCBUILDINGELEMENT	IFCDOOR	1
IFCDOOR	IFCDOOR	0

Figure 71 - Flattened IFC Object Hierarchy in BIMRL

7.2.1.7 Combining relational syntax and IFC objects

BIMRL uses a hybrid relational and object syntax to wrap internal mapping from IFC objects to the relational tables. Instead of specifying a record oriented statement such as "*Select ElementID from BIMRL_ELEMENT where elementtype='IFCDOOR' ...*", in

BIMRL the statement may look simply like "*CHECK IfcDoor D where Property(D, Reference, Pset_DoorCommon) like 'DOOR_P1_%'*".

7.2.1.8 Parameterization of rules

BIMRL has a concept of variables (described in more detail in section 7.2.2.1.4). Combining the variable with BIMRL triplet statements provides a means to parameterize the rule. For example instead of defining a statement with an explicit value:

```
CHECK
      { IfcSpace SA, BIMRL_TOPO_FACE F
        Where (PROPERTY(SA,OccupancyNumber)>50
```

The statement can be rewritten using a variable:

```
SETVAR ?occNo := 50;
CHECK
      { IfcSpace SA, BIMRL_TOPO_FACE F
        Where (PROPERTY(SA,OccupancyNumber)>?occNo
```

Using a variable to define the rule allows reuse of the rule with different values of the parameter without changing the rule. The variable can be changed during runtime at any time before the rule is executed.

7.2.2 Language elements

7.2.2.1 Basic Elements

7.2.2.1.1 Value types

BIMRL supports the basic data types: Real numbers, String, Boolean and Null.

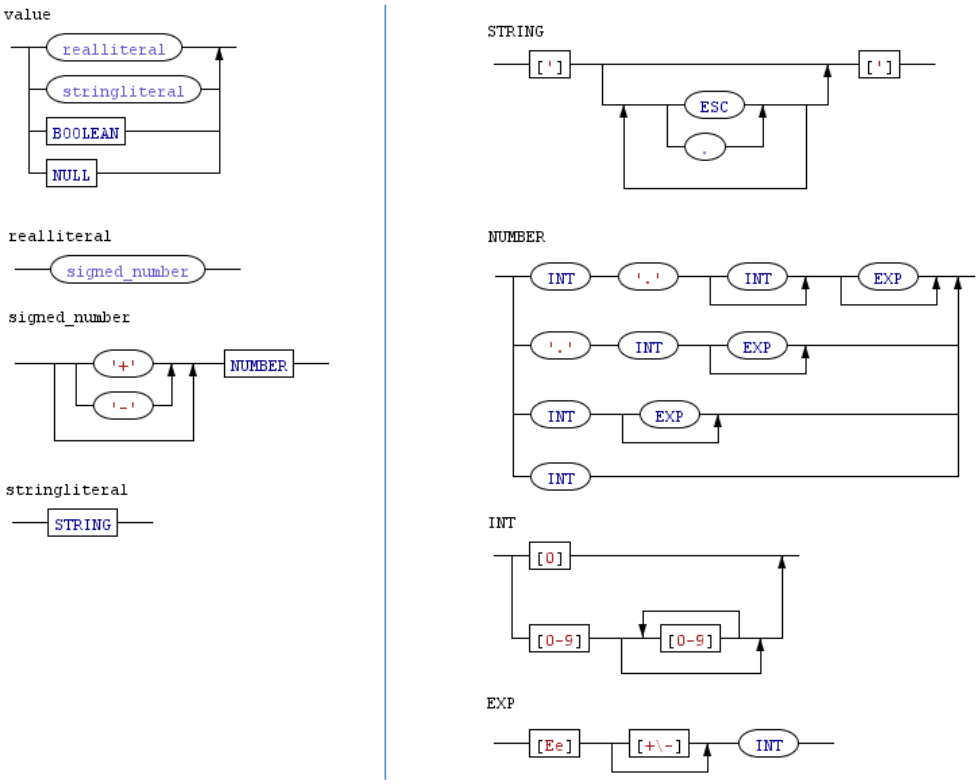


Figure 72 - BIMRL Basic Value Type

7.2.2.1.2 Alias

Alias is used widely in BIMRL to simplify the rule statement. Alias can be used in various places including table name, column name, geometry, set, etc. It generally takes a form of a single letter, but it can be any number of letters. In the following example "E" is an alias to the IFC types specified before it:

```
CHECK (ifcStair, IfcRamp, IfcSlab) E
```

7.2.2.1.3 IFC object types

There are two object types supported in BIMRL. One is the standard RDBMS object, typically a table name. The other one is a spatially processed IFC object. In BIMRL rule specifications, reference to an IFC object can be done directly to any of the IFC object within the hierarchy beginning from IfcProduct down, including the abstract types. For example:

```
CHECK IfcElement E …
```

or

```
.. WITH BACKGROUND (IfcWall, IfcStair, IfcStairFlight)
```

7.2.2.1.4 Variable

BIMRL supports user-defined variables that have a lifetime during the session. The variable can be used to predefine fixed values to be used in the main rule definition. The variable may take an explicit value, a value from an SQL query (from a table), and deferred value that is dependent on the bind variable that will be evaluated during runtime.

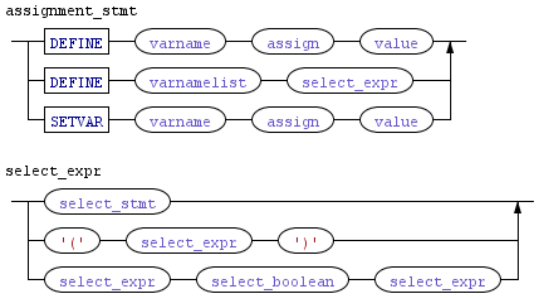


Figure 73 - Variable Assignment

The variable name takes a form of "?name". Valid examples for a simple variable assignment are:

```
DEFINE ?passValue := 0.5;
DEFINE ?buidlingType := 'PURPOSE GROUP V';
DEFINE ?createTempTable := .T.;
```

A set of values can also be set from the select expression that is an expression from a set of sql select statements. For example:

```
DEFINE ?max_distance, ?max_occupancy SELECT MAXDISTANCE,
  MAXOCCUPANCY FROM FIRECODE_TABLE_2_2_A
  WHERE BUILDINGTYPE = 'PURPOSE GROUP V';
```

There are several variables predefined in BIMRL mostly for the purpose of debugging. They influence the execution of rules by BIMRL during runtime. The reserved variables and their default values are:

- ?CreateTempTable (default value .TRUE.)

  BIMRL generates intermediate relations (tables) during runtime. By default they are created as temporary tables in the Oracle database, which is more efficient since they do not create a redo log. The downside of the temp table is they are only visible within the session. Changing the variable value to .FALSE. will alter BIMRL to create regular tables instead. These tables can be inspected outside of the BIMRL session.

- ?debugSQLStmt (default value .FALSE.)

  There are many SQL statements that are dynamically constructed during runtime. To see what may go wrong with certain SQL statements, this variable may be set to .TRUE., which tells BIMRL to write the statements into a log file.

- ?BackgroundTransparency (default value 0.5)

  To avoid the background elements in the exported X3D blocking the main objects that are of interest, the background model will be set to 50% transparent by default. Changing the value of this variable will alter the transparency value. Valid value is $0.0 \leq ?BackgroundTransparency \leq 1.0$.

7.2.2.1.5 Expression

An expression is a very useful concept in the SQL language. It supports nested expressions that allows for the definition of complex expressions. BIMRL supports some forms of expressions as follows:

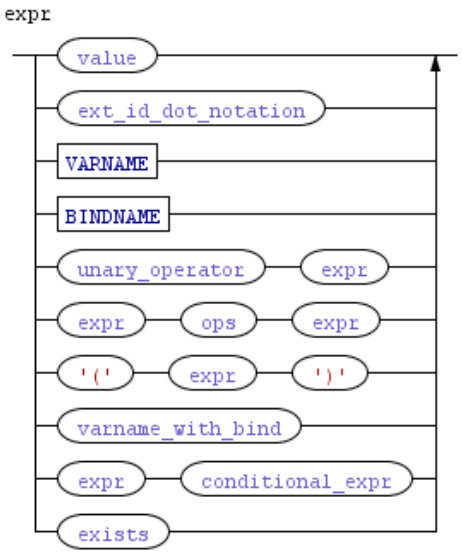


Figure 74 - BIMRL Expressions

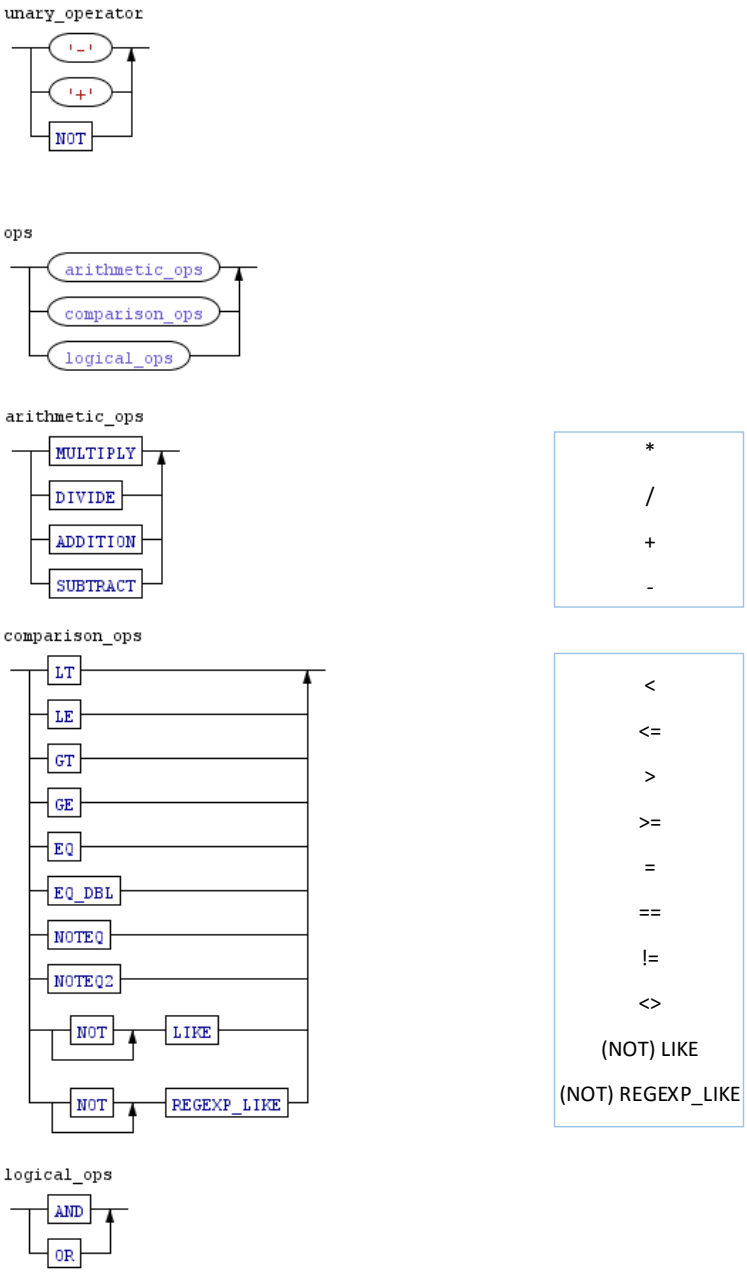


Figure 75 - Expression Operators (part 1)

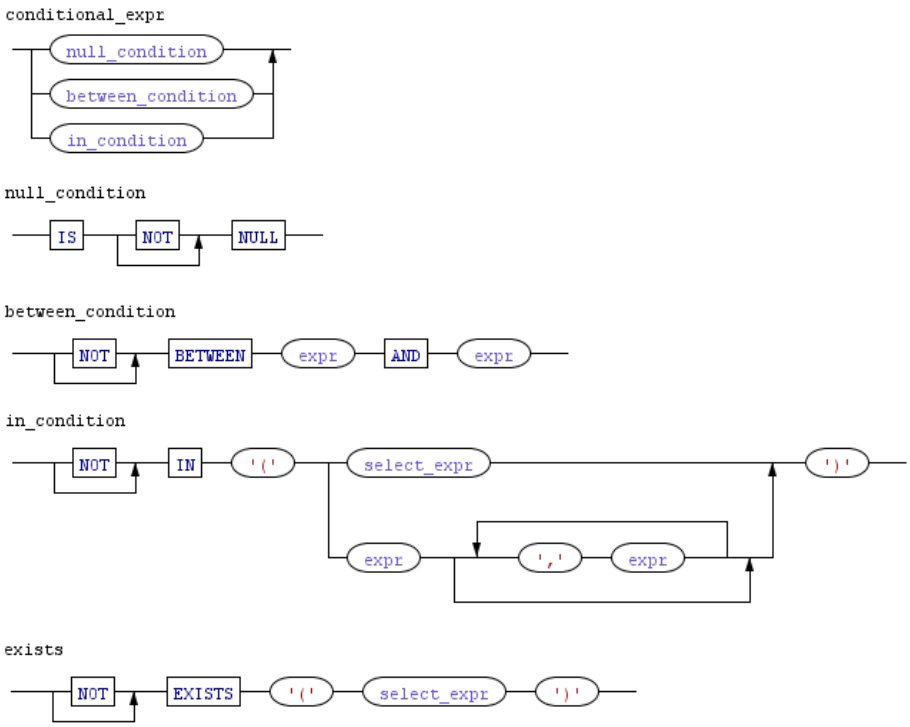


Figure 76 - Expression Operators (part 2)

7.2.2.1.6 Function

There are two types of functions supported in BIMRL. One type is the built-in function that provides many shortcuts, especially to various well-defined relationships in IFC. The other is the evaluation function that is mainly the extension function that supports plug-in architecture. The extension function is usually written to perform specific logic for rule checking, especially for the complex rules that require specialized algorithms. BIMRL has several built-in extension functions that perform frequently used generic functions such as intersection.

The first group of functions (built-in functions) are:

- AGRREGATEOF(alias) or AGGREGATEOF(M, E)

  Returning a reference to the element(s) which is aggregated to the element represented by the “alias”. This function has a variant that returns TRUE or

FALSE if given two arguments, where M is the master element and E is the aggregated element. It returns TRUE if the two objects are related by aggregation.

- AGGREGATEMASTER(E)

  It returns the master element which the element “E” is aggregated into.

- BOUNDEDSPACE(E) or BOUNDEDSPACE(E, S)

  It returns the space, which the element “E” is the boundary of. Similar to AGGREGTEOF, it also has a variant that returns TRUE or FALSE depending whether element E is a boundary of space S.

- BOUNDARYINFO(S, E)

  It returns information regarding the space boundary of element “E” that has a relationship to the specified space object “S”

- CLASSIFICATIONOF(E)

  It returns classification information assigned to the element “E”. The function returns the classification assigned to the instance or to the type, unless it is specifically specified in the argument using pre-defined qualifiers: CLASSIFICATIONOF(E, INSTANCEONLY), CLASSIFICATIONOF(E, TYPEONLY), or CLASSIFICATIONOF(E, INSTANCEORTYPE).

- CONNECTEDTO(E) or CONNECTEDTO(E1, E2)

  It returns a reference to the connected element to “E”. It can be used to trace the system connectivity or wall connection. If the second argument is supplied with another element alias, it returns TRUE or FALSE depending whether they are connected.

- CONTAINER(E) or CONTAINER(E,Y)

  It returns a reference to the spatial container element of the element “E”. If the second argument is supplied, it will check whether element “E” is inside the container “S”: CONTAINER(E, S). This functions works across the containment hierarchy, i.e. if an element “E” is physically contained in a space “S” and the

space “S” is an aggregate of a buildingstorey “Y” and “Y” in turn is an aggregate of a building “B”, CONTAINER(E, Y) or CONTAINER(E, B) will return TRUE. It is possible to restrict the traversal to only one level by restricting it using a WHERE clause, for example:

```
CONTAINER(E, S, 'WHERE LEVELREMOVED=1')
```

- CONTAINS(S) or CONTAINS(S, E)

  It returns references to the objects contained inside the spatial element S. By default it will be relying on the IFC containment information. This information may not be complete due to separation of models in linked files that do not share the same spatial structure. The “USEGEOMETRY” option can be specified in the parameter that will instruct BIMRL to use the spatial indexing information to perform the query. If the second parameter contains an alias, this function returns TRUE or FALSE depending whether a spatial structure element “S” contains element “E”.

- DEPENDENCY(M) or DEPENDENCY(M, E)

  It returns a reference to the dependent objects of element “M”, for example all openings in a wall, or door that fills an opening. The second variant returns TRUE or FALSE depending whether E is a dependent element of “M”.

- DEPENDENTTO(E) or DEPENDENTTO(E, H)

  It returns a reference to the host object of “E”, or returns TRUE or FALSE in the second variant if element “H” is the host of element “E”.

- ELEMENTTYPEOF(E)

  A simple function that returns an element type of element “E”.

- GROUPOF(E) or GROUPOF(E, G)

  It returns a reference to the group that the element “E” belongs to. In the second variant, it returns TRUE or FALSE depending whether group “G” is the group of element “E”.

- HASCLASSIFICATION(E)

  Returns a status TRUE or FALSE when element “E” has or does not have a classification assigned to it. Qualifiers INSTANCEONLY, TYPEONLY, or INSTANCEORTYPE (default) are applicable to this function.

- HASPROPERTY(E, <property name>) or HASPROPERTY(E, <property name>, <property set name>)

  This function checks whether the property “property name” exists in an element “E”. Without “property set name” it searches the existence of the property in all property sets that are defined for the element. Qualifiers: INSTANSEONLY, TYPEONLY, or INSTANCEORTYPE (default) are applicable for this function.

- MATERIALOF(E)

  It returns a reference to the material information of an element “E”. Qualifiers: INSTANCEONLY, TYPEONLY, or INSTANCEORTYPE (default) will determine which information will be searched.

- MODELINFO() or MODELINFO(E)

  This function returns the reference to model information. Without an argument, it returns all references to the models within the federated model. If element “E” is specified, it only returns the particular model that element “E” originates from.

- OWNERHISTORY(E)

  This function returns a reference to the IfcOwnerHistory information for the element “E”.

- PROPERTY(E, <property name>) or PROPERTY(E, <property name>, <property set name>)

  This function returns the reference to the property specified by the “property name”. Without “property set name” it searches for the existence of the

property in all property sets that are defined for the element. Qualifiers: INSTANSEONLY, TYPEONLY, or INSTANCEORTYPE (default) are applicable for this function. BIMRL keeps the property values in a generic string format to simplify search, but in some cases a simple cast from a string value to a number value or the other way round may be needed. To allow the cast, qualifier parameter TO_NUMBER and TO_CHAR can be specified.

- SPACEBOUNDARY(S)

  It returns a reference to the objects that provide the boundaries to the spatial structure element “S”.

- SYSTEMOF(E) or SYSTEMOF(E, Y)

  This function returns a reference to the SYSTEM that the MEP element “E” is a member of. Specifying the second argument will cause the function to return TRUE or FALSE depending whether element “E” belongs to the system “Y”.

- TYPEOF(E) or TYPEOF(E, T)

  This function returns a reference to the IfcTypeObject information of the element “E”. It returns TRUE or FALSE if the type alias “T” is specified in the second argument as the specific type of “E”.

- UNIQUEVALUE(E, <attribute name>) or UNIQUEVALUE(E, <property name>, (opt) <property set name>)

  This function returns a count of the existence of unique values in the element “E”. Valid attribute names are the following IFC direct attributes: NAME, LONGNAME, DESCRIPTION, OBJECTTYPE, TAG. It also accepts “property name” and optionally “property set name” if the property to be checked is of a specific property set. The element alias “E” can be a composite element from various element types depending on the overall query.

- ZONEOF(S) or ZONEOF(S, Z)

It returns reference to the IfcZone the space “S” belongs to. It returns TRUE or FALSE if the second argument is supplied with an alias to the IfcZone.

The second group of functions must adhere to the interface definition to be qualified as a plug-in. The base class ExtensionFunctionBase() provides basic operations to perform parsing of arguments, access to the checking result table, and an InvokeRule() stub function which by default will do nothing unless it is overridden. A few extension functions described below have been written in this research for some of the common functions frequently needed in rule checking. It is expected that the functions will grow over time, first with highly re-usable functions and followed by specialized functions. There is no hard rule to define the boundary as to what granularity the function should be defined, and it becomes a decision that must consider the trade-off between degree of re-usability and complexity of the usage. The higher the granularity of the function, the more complex the rule that uses it. Lower granularity on the other hand simplifies the rule definition, but makes it harder to re-use.

- Nothing()

  It is a dummy function that does not do anything. It is useful for executing rules that can be satisfied with the CHECK query alone.

- ComputeIntersection(E, B)

  ComputeIntersection essentially performs intersection operations between sets represented by alias E and B: $\phi_I = \phi_E \cap \phi_B$. This operation covers 7 of the 9-Intersection models, i.e. Contains, Inside, Equal, Touch, Covers, CoveredBy, Overlap. The complement of $\phi_I \rightarrow \phi_I^-$ will represent Disjoint. The first two parameters are mandatory. The first is the collection of the main object (with geometry) to check. The second parameter is the second set of objects that need to be checked to see whether there is any intersection.

There are additional parameters that the function accepts, such as the "EXCLUDEID" sub-clause to list column alias in which the element ids are to be excluded in the intersection operation. This is useful to exclude the main objects that are being checked from the false positive that will be generated otherwise in some cases. Another parameter is "EXACT". Since the result of intersection ϕ_I does not discriminates between the 7 operations due to the approximation nature of the Octree decomposition, additional and more precise computation may be needed to identify the actual status of just one of the 9-Intersection models. The "EXACT" parameter qualifier will instruct the function to perform a second level check based on the actual geometries using the intersection set ϕ_I. While Octree based intersection is extremely fast, the precise intersection will be slower.

ComputeIntersection function also returns the output into a specialized temporary table USERGEOM_OUTPUTDETAILS where objects that are related to the intersection result will be listed. This information is useful to highlight what objects will be present in the visual output to X3D and their location.

- VolumeIntersection(E, S)

  This function offers a simple volume intersection of two solids. Only the Octree based intersection will be evaluated for an approximate volume intersection between the two solids.

- ComputeRemoteLocation(ID1, ID2, F, E, EC)

  This function is a specialized checking function that calculates direct distance between two openings compared to the diagonal distance of a set of objects in a building. This function can be used for two types of rules that check the distance of exit doors from an open office at a design stage when it is still empty, or a rule that checks the existence of remotely located exit doors from a specialized room such as a patient ward. Five parameters are required for this function:

- ID1 – the set of IDs from the first object that represent the room or floor
- ID2 – the set of IDs of the second object representing the openings
- F – alias that provide the face geometry information, typically the footprint
- E – an alias to the set of the second object
- EC – a location represented (centroid) of the second object

Each of the functions above generally accepts additional optional arguments for WHERE clauses as an additional condition or filter. Examples of the use of the functions above can be found below:

```
WHERE TYPEOF(E).ifctype in
  ('IFCSWITCHGEARTYPE','IFCTRANSFORMERTYPE')
COLLECT CONTAINER(E, "WHERE LevelRemoved=1 and
  SpatialElementType='IFCBUIDLINGSTOREY'").name STOREYNAME
COLLECT boundaryinfo(c2,d).commonpointats bp2
```

7.2.2.1.7 Geometry types

One of the important features in BIMRL is the ability to generate transient geometry during runtime to assist the rule checking. In most cases, the geometry is generated relative to the building element's geometry. For example, a box may be generated by an extrusion above the top face of an element's OBB, or a line is created by connecting a boundary face centroid of one space and another space sharing the same common space. Several geometry types supported are listed below:

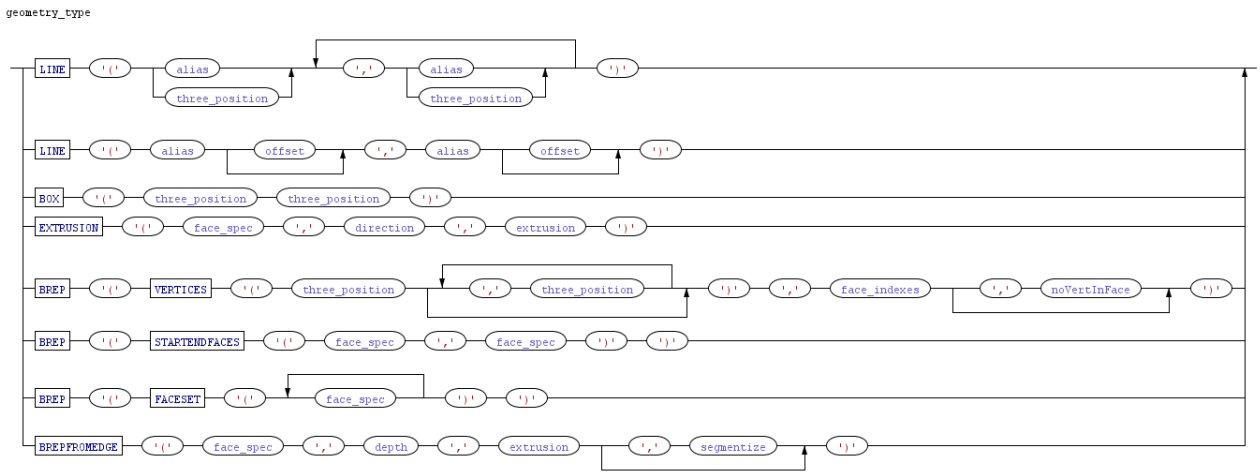


Figure 77 - Geometry Types that can be created for the Transient Geometry in BIMRL

7.2.2.1.8 Supported output format

The report is an essential part of any rule checking. A good report not only reports the final outcome of the rule check, but also provides additional information that help describe what is being checked and why certain design elements may fail a specific rule check. In the BIM rule context, an additional visual report will be very helpful to assist the user to identify what exactly the cause of the problem is and hence the right solution to overcome the issue. In BIMRL, there are currently two possibilities to capture the report:

- Text (table) based reports. BIMRL uses relations (tables) in processing the rule check. Therefore the outcome of the check can also be stored in a table. This includes the transient geometry information that may be of interest to the users.
- X3D visual report. BIMRL uses X3D format to export the geometry of objects that users are interested to view. It is not limited to failed objects only, but also to any of the checking result. The X3D export allows the user to specify only the relevant part of the model to be exported. The advantage of using the X3D viewer is that it is an open standard format that makes it portable. Furthermore, there are many available free X3D viewers that can be used to view the visual report.

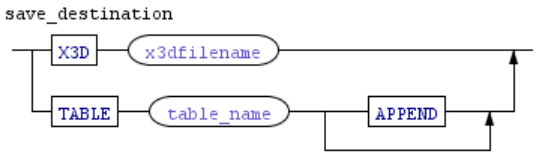


Figure 78 - Supported Output Formats in BIMRL

7.2.2.2 The BIMRL rule triplets

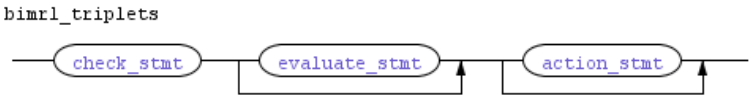


Figure 79 - BIMRL Triplets

The main part of BIMRL that allows the user to define rules and that will be executed during runtime to perform the actual check is a triplet command. The triplet is formed by three major sub-clauses: CHECK, EVALUATE, and ACTION. They embody the general structure of building rules that in general also closely resembles the NRL format.

7.2.2.2.1 CHECK

The main role of the CHECK sub-clause is to perform a selection or filter for the main object of interest in the rule. This represents the selection of the main objects of interest and also applying their constraints as specified in the CG. Figure 80 shows the main syntax and language elements involved in the CHECK sub-clause.

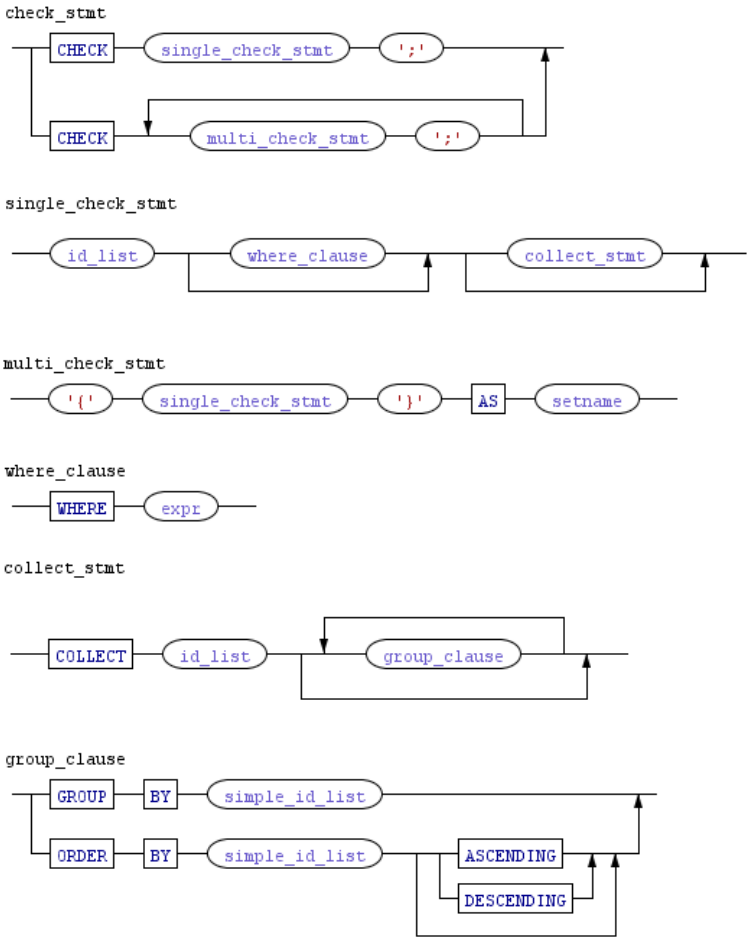


Figure 80 - CHECK statement

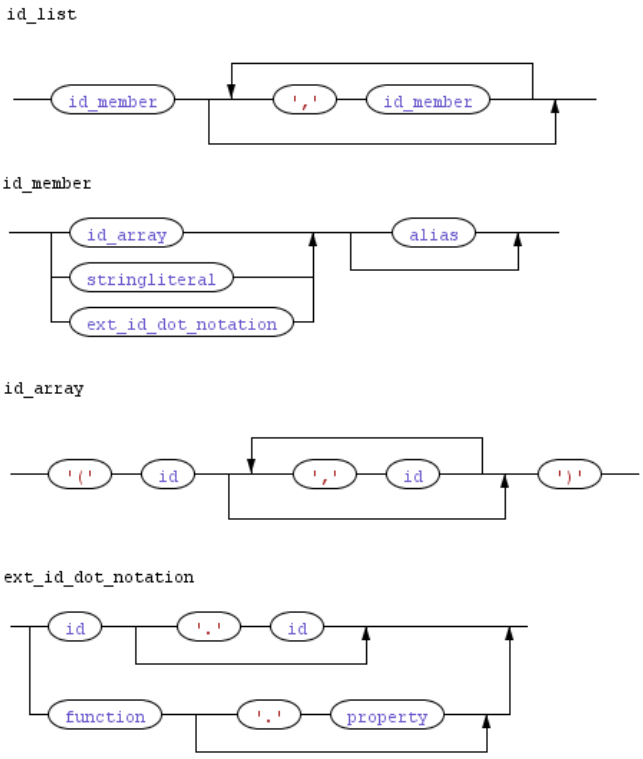


Figure 81 - ID List details for CHECK statement

The CHECK statement is similar with the SQL Select statement. At the end of the parsing, the CHECK statement will be converted into an SQL statement too. The id list at the beginning of the CHECK statement lists (Figure 81) either a table name or an IFC object (see the previous section 7.2.2.1.3 for details). The use of aliases is recommended to simplify the rest of the statement when making references to the object. IFC objects can also be grouped into a collection with a single alias.

CHECK statements can take the form of a single select statement or multiple select statements resulting in different sets (tables). The sets will be used in the evaluation stage.

The COLLECT sub-clause specifies columns and aliases given to the columns that will provide the column projections in the final parsed SQL statement. As mentioned, the result of the CHECK statement will be created also as a relation (table)

temporarily during the execution of the rule. Figure 82 shows an example of a BIMRL CHECK statement and its final translation into an SQL Statement.

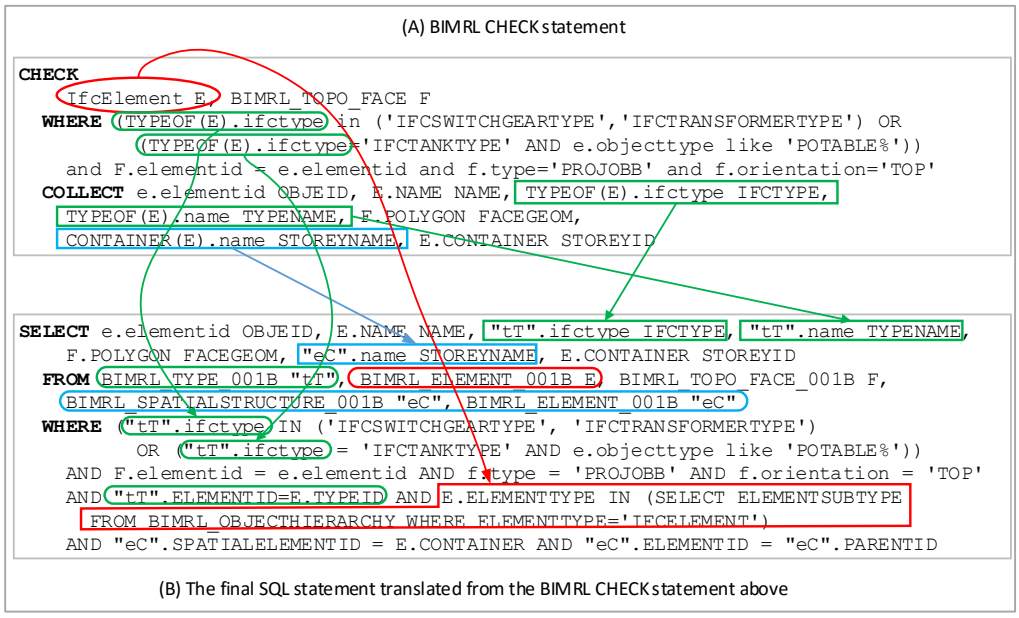


Figure 82 - An Example of BIMRL CHECK Statement and Its Translation to an SQL Statement

7.2.2.2.2 EVALUATE

Evaluate statements are where the actual checking logic should occur. It works in tandem with the CHECK set(s). Figure 83 shows the syntax of EVALUATE statement.

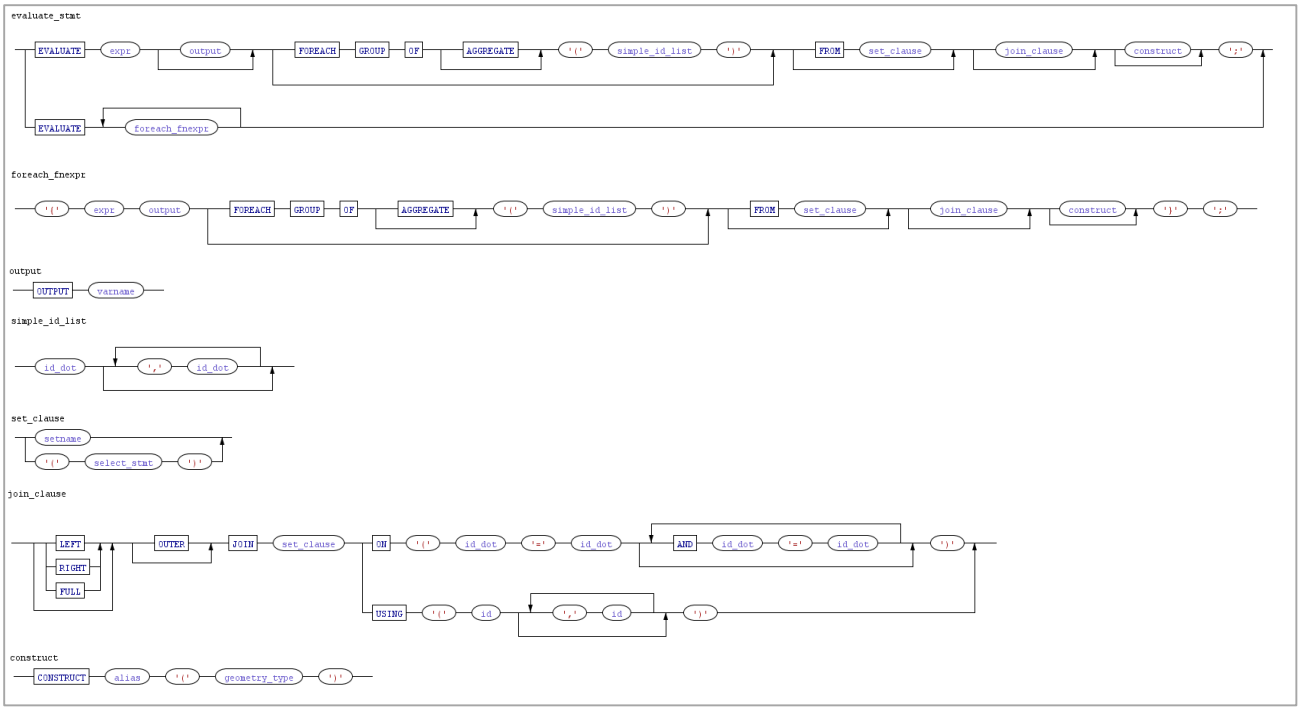


Figure 83 - EVALUATE Statement

EVALUATE statement sets created by CHECK statements can be joined before it is passed into the Evaluate function (within the expr clause). The aggregate keyword is an optional keyword that tells Evaluate if the join result should be aggregated (grouped) by the specified columns. At the same time, the Construct sub-clause is also called in this statement if transient geometry should be created and used for the Evaluation function. The Evaluate function does the heavy lifting for executing checking logic, while the rest of the statement provides support to prepare for that. Figure 84 shows an example of the EVALUATE statement and the explanation as to what actually happens in the background.

```
CHECK
    { IfcElement E, BIMRL_TOPO_FACE F
      WHERE (TYPEOF(E).ifctype in ('IFCSWITCHINGDEVICETYPE','IFCTRANSFORMERTYPE')
             OR (TYPEOF(E).ifctype='IFCTANKTYPE' AND e.objecttype like 'POTABLE%'))
        and F.elementid = e.elementid and f.type='PROJOBB' and f.orientation='TOP'
      COLLECT e.elementid OBJEID, E.NAME NAME, TYPEOF(E).ifctype IFCTYPE, TYPEOF(E).name TYPENAME,
        F.POLYGON FACEGEOM, CONTAINER(E).name STOREYNAME, E.CONTAINER STOREYID
    } as SET1;
    { IfcDistributionElement D
      WHERE SYSTEMOF(D).objecttype like 'Sanitary%'
      COLLECT D.ELEMENTID MEPOBJEID, D.name MEPOBJNAME, SYSTEMOF(D).name SYSTEMNAME
    } as SET2;

EVALUATE
        COMPUTE INTERSECTION (EB, SET2.MEPOBJEID) OUTPUT ?clash FROM SET1
        CONSTRUCT EB (EXTRUSION (defface(FACEGEOM), +ZAXIS, 15));
```

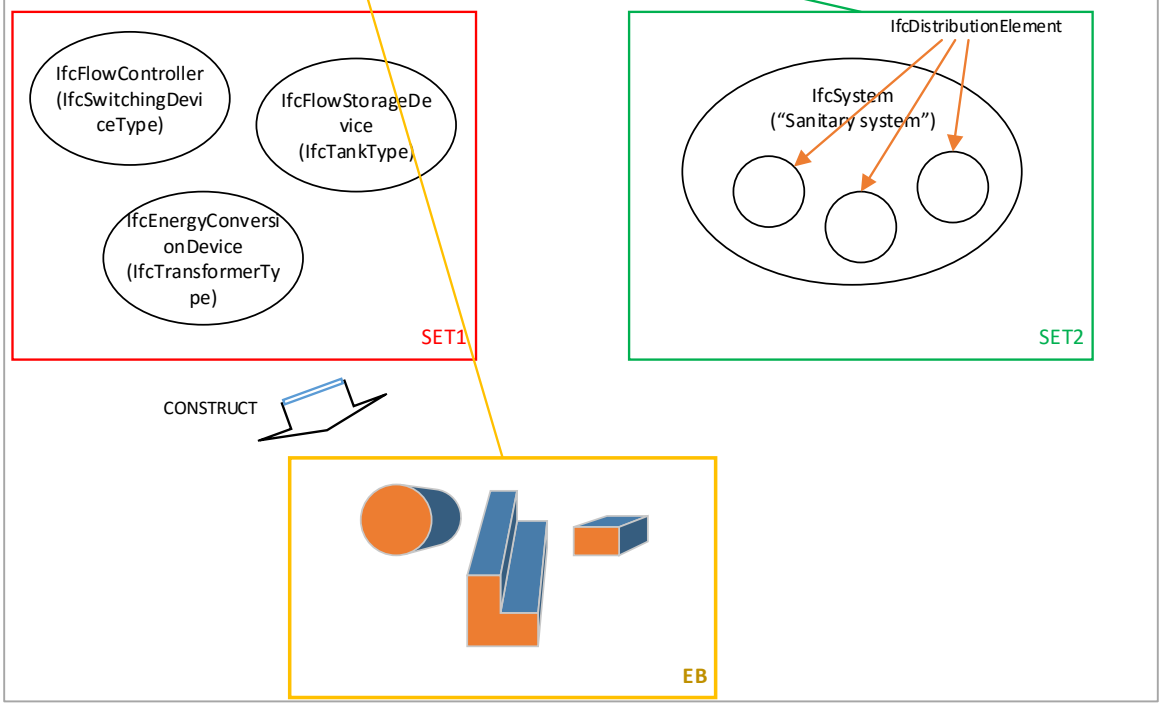


Figure 84 - An Example of BIMRL EVALUATE Statement

Output keywords lets users specify an internal variable that will be used to generate the temporary table for the result of the Evaluate function. It also serves as an internal variable the holds the checking result in a column named OUTPUT.

- Chained Evaluation

  For some complex rules, there may be an evaluation function that must be executed only after another evaluation function is completed. It uses the input from the previous evaluation function to perform the checking logic. In this case

the evaluation functions need to be “chained” to achieve the effect. Figure 85 shows how that can be achieved using multiple EVALUATE sub-clauses.

```
EVALUATE
  { COMPUTEINTERSECTION(EB, SET2.MEPOBJEID) OUTPUT ?clash FROM SET1
      CONSTRUCT EB (EXTRUSION (defface(FACEGEOM), +ZAXIS, 15))
  };
  { COMPUTEINTERSECTION(XB, MEPOBJID) OUTPUT ?protection
      FROM SET3 JOIN (SELECT * FROM CLASH WHERE OUTPUT>1)
      CONSTRUCT XB (EXTRUSION (defface(FACEGEOM), -ZAXIS, 10) Offset(0,0,1.5)
  }
```

Chaining

The final output

Figure 85 - Chaining Evaluation Functions in EVALUATE Statement

7.2.2.2.3 ACTION

Action statements lets users specify what action is to be performed out of the result from the evaluation function. Figure 86 shows the syntax for the ACTION statement, which may have just a single action or multiple actions using a WHEN sub-clause to split the appropriate actions.

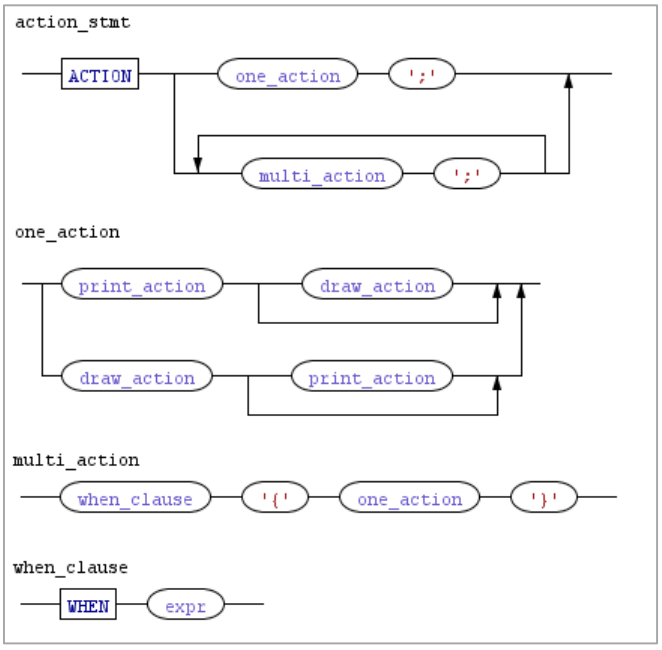


Figure 86 - BIMRL ACTION Statement

There are two major functions within the ACTION statement, i.e. PRINT and DRAW actions. The Print action will write the requested output to the BIMRL console in

a tabular format. It can be combined with save_action that provides three options to save the result to: Database tables or X3D format for geometry. The details have been described in section 7.2.2.1.8.

7.2.2.2.3.1 Print action

Print action is a straightforward action to print out the result into BIMRL console and to tell BIMRL to save the information into the supported a table (Figure 87). The output can be consolidated into one or multiple tables. The APPEND keyword should be used if the same table should be used and the result from one action sub-clause should be appended to it. Without the APPEND keyword the table will be overwritten.

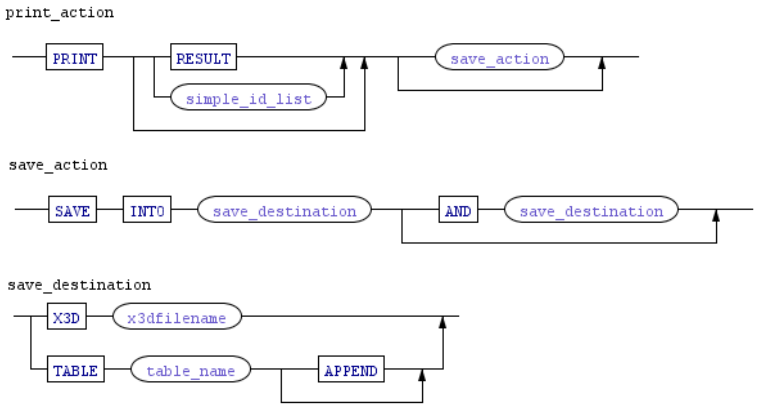


Figure 87 - Print Action

7.2.2.2.3.2 Draw action

Draw action has a number of important features that allows for the flexibility to select only relevant information to be generated into a X3D file. It is especially useful when background objects from the model is required to give clarity and completeness of the visual report. There are several important items that can be performed in the draw action (Figure 88):

- Draw the checking result geometry from the transient geometry with a specific color
- Selective output of background IFC objects

- Highlight specific objects using specific color
- Define transparency for the background model

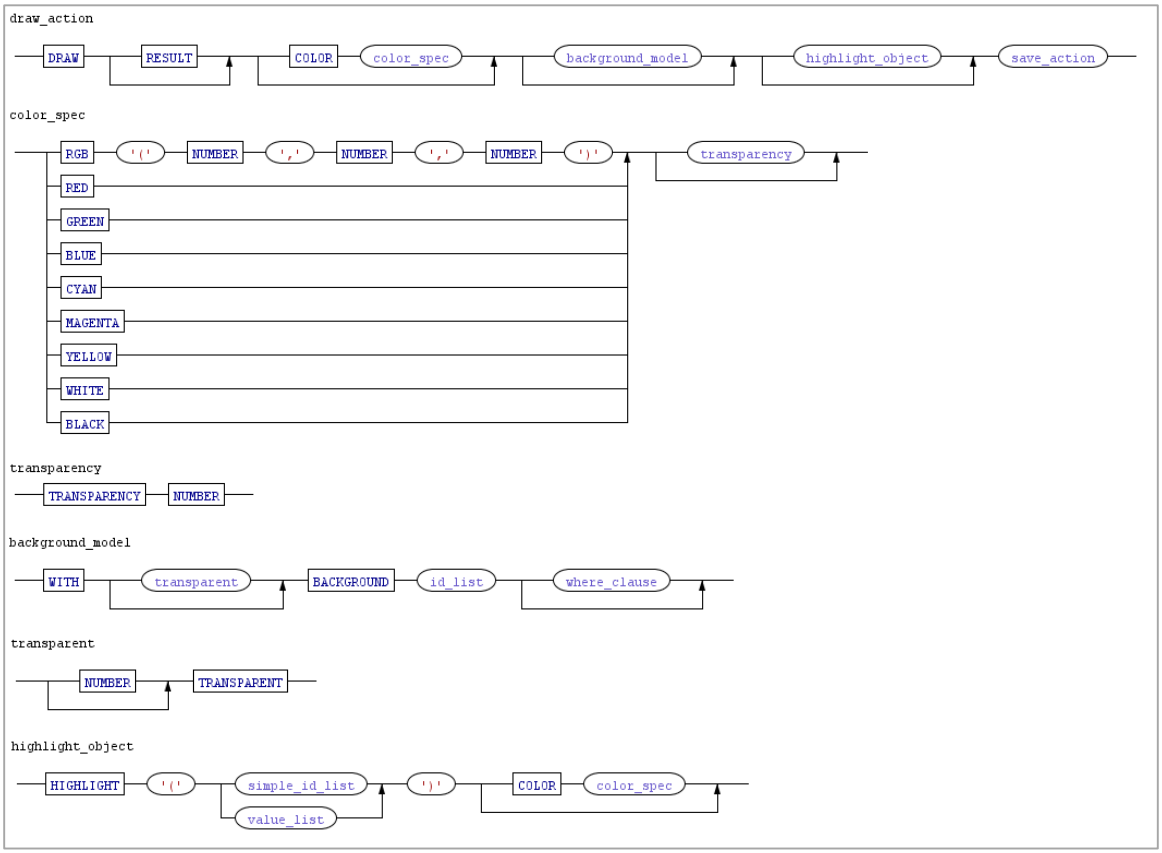


Figure 88 - Draw Action

7.2.3 Supporting commands

7.2.3.1 Commands related to a Variable

- DEFINE

  Defining a variable and assigning it with a value or assigning a variable to a query to a data source in a table

- SETVAR

  Command to set or update a variable value. If the variable name in SETVAR command is not yet defined, it will define it, and otherwise it will update the value.

- SHOW VARS

  Shows all variables defined within the session of BIMRL

- DELETE

  Delete a specific variable

- RESET VARS

  Delete all variables in the session

7.2.3.2 SQL pass-through.

This command allows an un-interpreted SQL statement to be processed and executed in the BIMRL interface. The statement is directly passed to the underlying SQL interface. The only “interpretation” done by BIMRL is to translate all BIMRL tables into the actual internal table names. This functionality is useful as an additional facility to access the underlying database features in addition to BIMRL. For example to manage tables, to query the data not related directly with BIMRL.

7.2.3.3 DELETE model

Delete a specific model in the BIMRL schema by its ID or project name and number.

The complete BIMRL language grammar definition in BNF format can be found in APPENDIX C. The design of the language is to define a rich enough functionality and with powerful set of relational algebra that allows for a wide range of queries. Even with this, the language will not be able to deal with every possible rule written. Therefore, the goal of BIMRL is to define a standardized structure (or grammar) of the language that is expected to be stable. What cannot be anticipated or not part of the standard can be added into the language without changing the grammar. In rule checking context, they are mainly additional evaluation functions. With the possibility to add extension functions,

the ability to define parallel evaluation functions and to define chained evaluation, BIMRL should provide a good platform for generic rule checking systems.

With a built-in support of geometry and graphs, BIMRL will be complete to handle complex rules [91]. Class-1 rules, most class-2 rules and some class-3 rules can be supported directly in BIMRL mostly with just one or a few rule definitions. It is not expected that BIMRL can solve all rules, nevertheless it has brought down the barrier of entry to complex rules using a combination of simplified database schema, standard geometry support in the database and standardized rule language. It has simplified many things in both horizontal (diverse of rules) and vertical (complexity of rules) directions. It makes the dream for automated rule checking to be much closer to reality.

CHAPTER 8

PROOF-OF-CONCEPT

8.1 Implementation Details

A prototype software has been written to test how well the concept described in this thesis performs. Two major pieces of this prototype are the ETL program that reads IFC file into Xbim and then into BIMRL Oracle schema, and the BIMRL language interface. The software is written using C# programming language. ANTRL is also used as the parser tool to parse BIMRL statements in the BIMRL language interface. The diagram in Figure 89 shows the software architecture for this prototype.

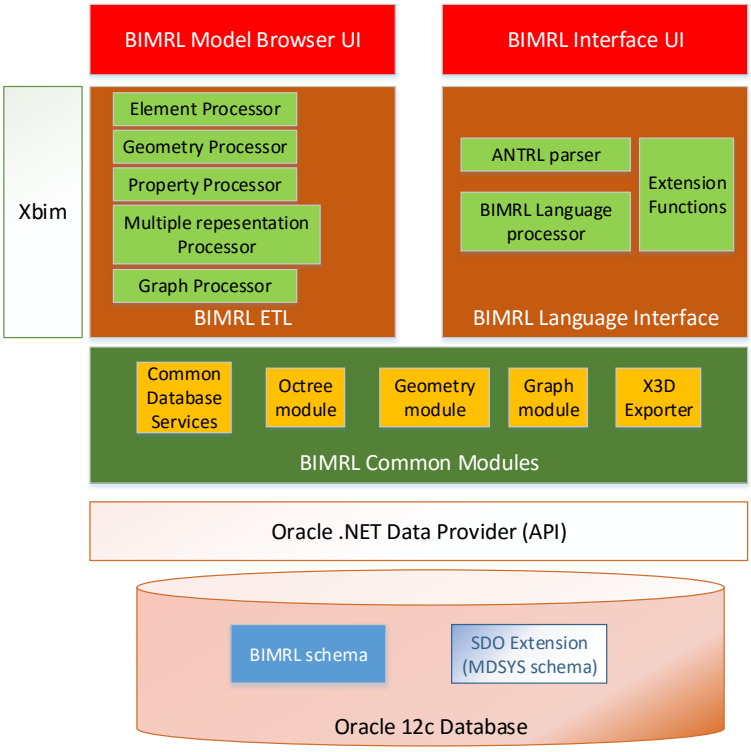


Figure 89 - Software Architecture of the BIMRL Prototype Implementation

8.1.1 BIMRL ETL Module

The ETL process is currently a simple linear process that consists of multiple steps (Figure 90). There are three distinct steps, one occurs outside of BIMRL using Xbim and the other two occur inside BIMRL. They can be run as a single-click automated process from the Xbim Xplorer interface (Figure 91).

1. Importing IFC(s) into Xbim. Xbim allows creation of a federated model by appending set of related IFC files.
2. "Pushing" the data from Xbim into the BIMRL database schema that involves creating the BIMRL tables, processing IFC elements, element geometry as a polyhedron data from the triangulated mesh, and element properties and the relationship.
3. Processing the multiple representation data inside the BIMRL database. This step involves four distinct steps:
    a. Processing Boundary Representation Faces from the "triangle soup" of the polyhedron geometry.
    b. Computing OBB and projected OBB
    c. Computing the simple AABB
    d. Computing the Octree decomposition or indexes

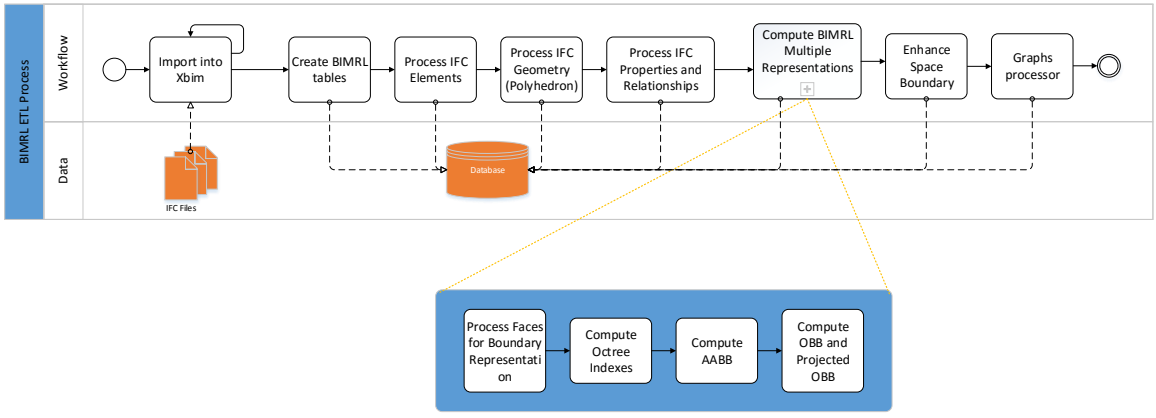


Figure 90 - The BIMRL ETL Process Workflow

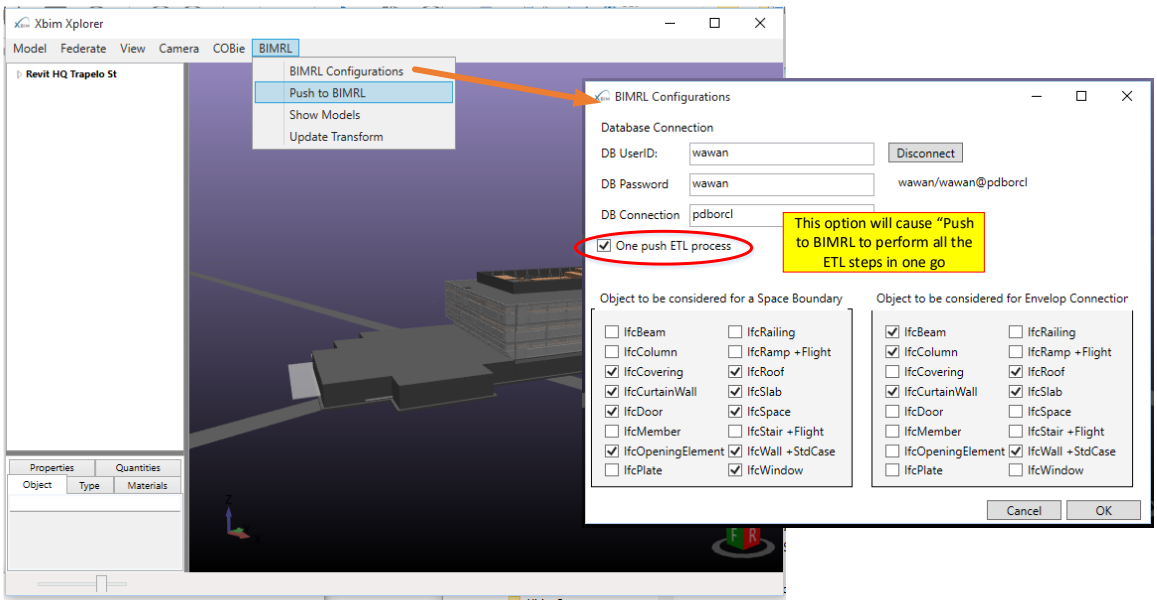


Figure 91 - One Push ETL Process from Xbim Xplorer Interface

The various steps in the ETL process can also be repeated individually through the BIMRL model browser UI. It is a very handy tool in the course of the research to study the effects of changing parameters to influence the computation such as changing the Octree maximum level of subdivision, or investigating errors, and updating selected data when some parameters are tweaked, e.g. tolerance (Figure 92).

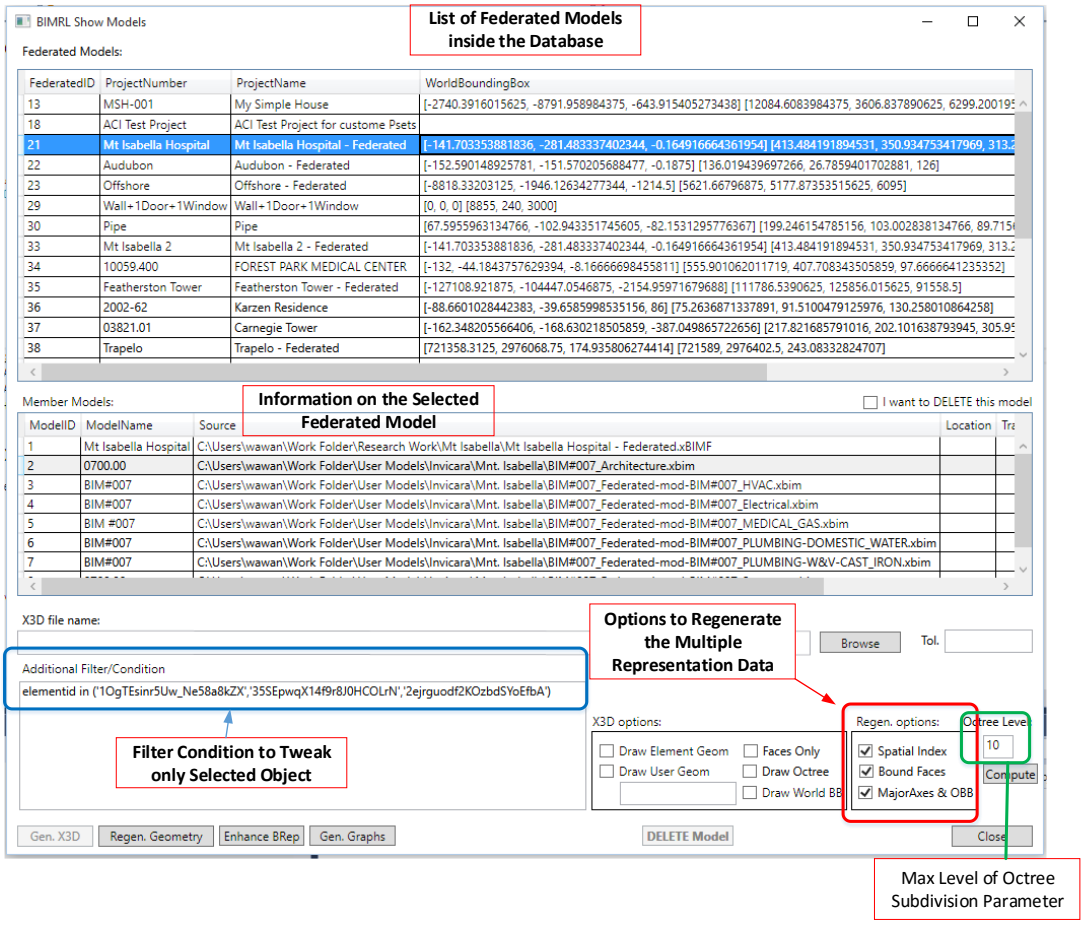


Figure 92 - BIMRL Model Browser UI

8.1.2 The BIMRL Language Interface

The BIMRL Language interface is a standalone UI based application written also using C# programming language and with XAML UI. It accepts BIMRL grammar and allows to test the grammar for its validity by just parsing it, or it can also execute it. The result of the execution is displayed in the UI under the result panel. Variables that are defined within the session will be displayed in the special panel on the right (Figure 93).

The language parser is implemented using ANTRL (ANother Tool for Language Recognition) version 4 [150]. It is an open source tool that provides the parsing mechanism and generates walk parse tree or AST (Abstract Syntax Tree). The parsing option in the UI basically walks the statement and generates the AST. In the execution

mode, it does two steps operations, first it parses the statement into an AST, and second, it walks the tree and executes the interface implementation for BIMRL. There are many other parsers available, but ANTLR is selected due to its mature implementation, advanced features, supporting tools, active community, support for different target language that includes C#, and the availability of a good documentation that often is a weakness in any open source project.

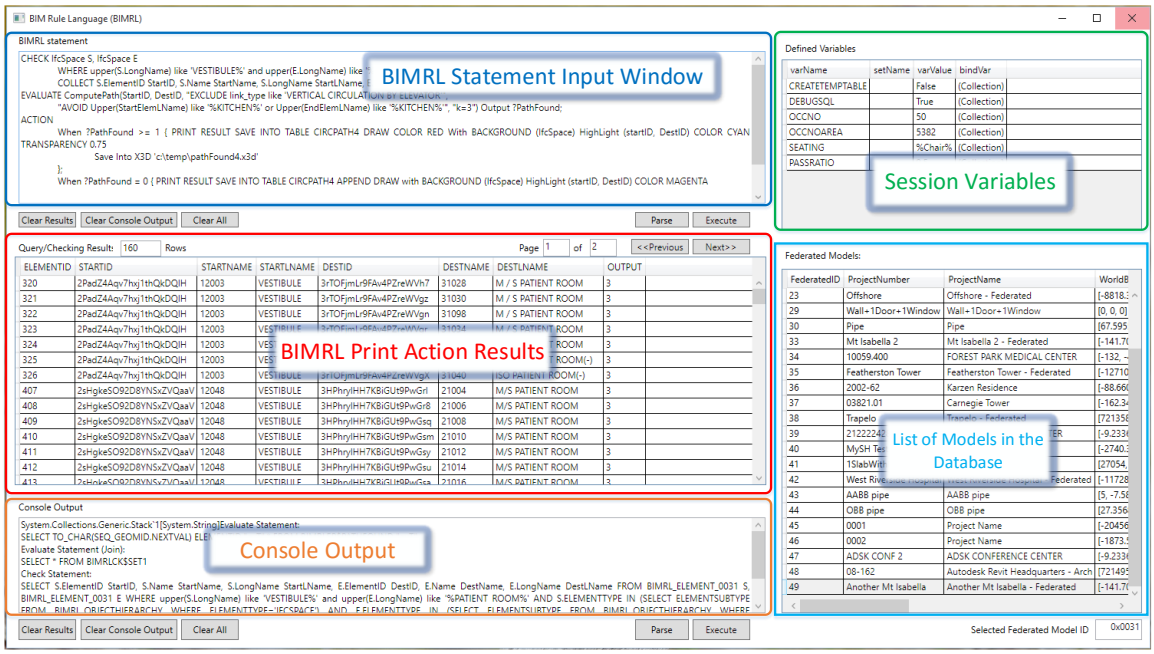


Figure 93 - BIMRL Language User Interface

To demonstrate the proof-of-concept, a few BIM related rules are selected. Since there is no well-defined classifications of rules from the perspective of the type of rules or checking requirements exists, the rules selected are based on rules that will exploit the key concepts introduced in this research. The selected rules are also selected based on the availability of the data set, which are the realistic models (see APPENDIX B for the details of the models). The details descriptions for each of the selected rules and their associated BIMRL statements are presented in the next sections.

8.1.3 Language Extension

With a virtually infinite number of possible rules, it is not possible for a language to be implemented to cover them all. Therefore a concept of extension is built into BIMRL. From the breakdown structure of the rules (CHAPTER 4), it is evident that most extension will be required in the area of checking functionality. With this, BIMRL allows extension to the evaluation function. The extension function should be written by deriving a class from the base function and the interface:

```
public class ComputeIntersection : ExtensionFunctionBase,
                                   IBIMRLExtensionFunction
```

The Interface IBIMRLExtensionFunction defines at least one function that is typically overridden, i.e. `public override void InvokeRule(DataTable inputDT, params string[] inputParams)`. It takes two arguments: the first is a DataTable which is the result of a Join statement within the EVALUATE section, or the set from CHECK statement if there is no Join statement specified. The second argument is a parameter list that should be function specific. The specifications should be documented so that the user of the function knows what information should be provided when calling the function. A few standard parameters or qualifiers can be defined. They are assigned into dedicated variables inside the base function.

- WHERE parameter. A where parameter can be used to add additional filter for the extension function to process. It is an additional SQL where clause to add control for the extension function to apply additional specific filter.
- EXCLUDEID parameter. A special parameter that is followed by a list of element ids that should be excluded in the process within the extension function.
- AGGREGATE parameter. A special parameter that specifies the list of columns to be used as an aggregate condition inside the function.
- Function qualifiers that specifies a specific predefined option. The list of predefined enumeration for the function qualifiers are:

```
public enum functionQualifier
{
    INSTANCEONLY,           // Evaluate respective information
                            //   pertaining to Instance only
    TYPEONLY,               // Evaluate respective information
                            //   pertaining to Type only
    INSTANCEORTYPE,         // Evaluate respective information both
                            //   from Instance and Type
    USEGEOMETRY,            // Evaluate using Geometry for spatial
                            //   operation
    PHYSICALBOUNDARY,       // Specific for Space boundary:
                            //   Physical
    VIRTUALBOUNDARY,        // Specific for Space boundary: Virtual
    EXTERNAL,               // Is external
    INTERNAL,               // Is internal
    COMMONPOINT1,           // a common point of boundary face
                            //   between 2 elements
    COMMONPOINT2,           // a common point of boundary face
                            //   between 2 elements
    FACEID1,                // a face id to the first element in
                            //   the argument list from boundary
                            //   face
    FACEID2,                // a face id to the second element in
                            //   the argument list from boundary
                            //   face
    TOP,                    // Face orientation: TOP
    BOTTOM,                 // Face orientation: BOTTOM
    SIDE,                   // Face orientation: SIDE
    TOPSIDE,                // Face orientation: TOPSIDE
    UNDERSIDE,              // Face orientation: UNDERSIDE
    EXACT,                  // To tell function to operate on exact
                            //   geometry and not only its
                            //   approximated shape using the
                            //   spatial index
    AGGREGATE,              // To tell the function to consolidate
                            //   the result table into a unique
                            //   aggregate columns as specified
    TO_NUMBER,              // Change the value to number format
    TO_CHAR,                // Change the value to string
    USE_OBB,                // Option to use Projected OBB, without
                            //   this option, the default will be
                            //   AABB
    UNDEFINED
}
```

The base function ExtensionFunctionBase does the standard processing for the function including keeping the above parameters in the dedicated variables, checking for special parameters related to the creation of the transient geometry, and a standard method for BIMRL to access the final result from the function. One additional mandatory item the extension function must provide is an additional column OUTPUT to keep the evaluation result. This column values are used in the ACTION section WHEN clause represented by the evaluate output variable.

Several pre-built extension functions have been developed in this prototype. They are used in the test cases described in the following sections. The same approach can be used for various use cases. For example the use of a graph can be included in the database and the access to the graph related function such as shortest path can be provided through the extension function. Another interesting application that can be implemented the same way is to link BIMRL with an external simulation function. For example, an energy simulation program such as EnergyPlus, or a Leakage Analysis using CONTAM can be linked using the extension function. BIMRL will provide access to the BIM data and allow specific information from the graph to be generated and sent to the external simulation program. The extension function is responsible to collect the analysis results and process it into a table in BIMRL that can be used to create report. This process is illustrated in a diagram shown in Figure 94.

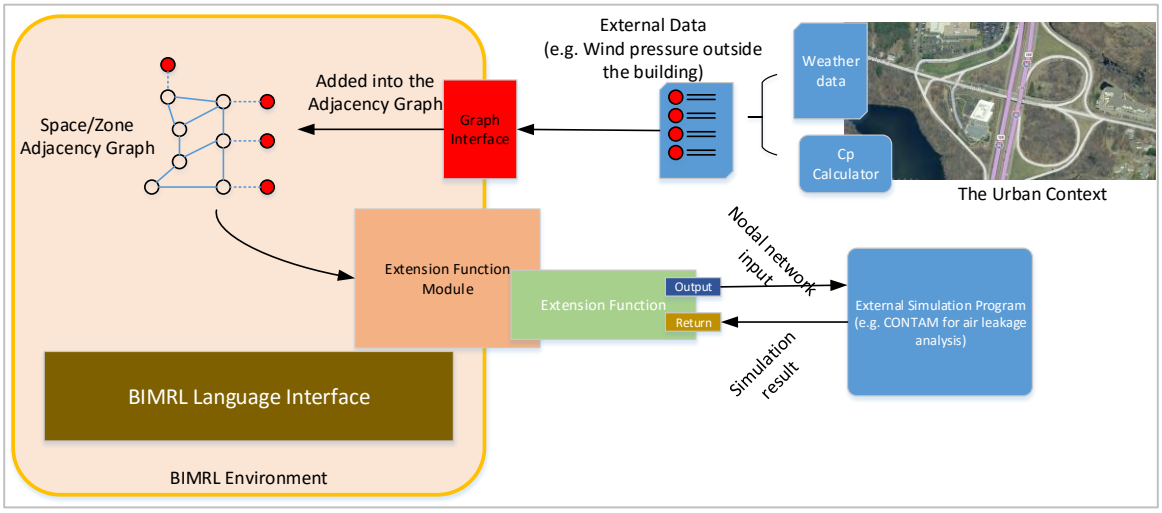


Figure 94 - Adding Extension Function to BIMRL with a Link to an External Simulation Program

8.1.4 Graph Support

For the purpose of this proof-of-concept, a simple support for graphs inside the database is implemented. The graph as discussed in CHAPTER 3 is also an important component often required for Class-3 rules. In this proof-of-concept, a simple circulation graph is generated as part of the ETL process. The graph is generated by connecting

Spaces with the walkable boundary objects in the horizontal direction, i.e. doors, openings, or another space that is directly connected. In the vertical direction, elevators, stairs and ramps are searched and vertical connections are generated as necessary. This prototype only implement the logical paths and not the actual physical walking path. Lee [1] has demonstrated integration of the actual walking path for the purpose of fire exit. Adding such data is simple in BIMRL since the path is just a simple line geometry that can be added into the graph data. In this proof-of-concept, the emphasize is to demonstrate the agility of integrated graph into the database and BIMRL environment since the graph can be altered dynamically to control what path is desirable and which is not. More detail description is found in Case-4 (section 8.5).

The internal storage of the graph nodes and links uses Oracle Network Data Model schema. A view has been created for a convenience combining the node, link, BIMRL element tables as followed:

Table 15 - A View for Circulation Graph Data

CIRCULATION_nnnn	*(nnnn is hexadecimal model id)*	
Column Name	Description	
LINK_ID	Graph link (edge) id	From Oracle Network Link table
LINK_NAME	Link name (ElementIDs of both ends of the link)	
LINK_TYPE	Type of the link	
ACTIVE	Status of the link	
LINK_LEVEL	Hierarchy level of the link	
PARENT_LINK_ID	Parent link id	
START_NODE_ID	Graph node id	From Oracle Network Node table (start node)
START_NODE_NAME	Start node name (= ElementID)	
START_NODE_TYPE	Start element type	
START_NODE_ACTIVE	Status of the start node	
START_HIERARCHY_LEVEL	Hierarchy level of the start node	
START_PARENT_ID	Parent id of the start node	
END_NODE_ID	End node name (= ElementID)	From Oracle Network Node table (end node)
END_NODE_NAME	End element type	
END_NODE_TYPE	Status of the end node	
END_NODE_ACTIVE	Hierarchy level of the end node	
END_HIERARCHY_LEVEL	Parent id of the end node	
END_PARENT_ID	Parent id of the end node	
STARTELEMENTNAME	Name attribute of the start element	From BIMRL_Element table (start node)
STARTELEMENTLNAME	LongName attribute of the start element	
STARTELEMENTOBJECTTYPE	ObjectType attribute of the start element	

ENDELEMENTNAME	Name attribute of the end element	From BIMRL_Element table (end node)
ENDELEMENTLNAME	LongName attribute of the end element	
ENDELEMENTOBJECTTYPE	ObjectType attribute of the end element	

8.2 Case 1: Hospital Design - Best Practice

"All patient rooms must be visible from the nurse station".

This rule can be expressed in a CG in a rather simple form (Figure 29). It requires 2 derived concepts, i.e. line of sight and access. This best practice has been identified to be one of the key safety design principles for a hospital building [53]. It has led into various design innovation to accommodate the need for this visibility [51-53]. For example, a widely published design of Miami Valley Hospital in Ohio, incorporates such thinking into the design (Figure 95) [151].

Figure 95 - An example of design incorporating patient room visibility into Miami Valley Hospital [151]

Figure 96 - Visibility Requirement from the Nurse Station to the Patient Rooms

The requirement states that there has to be a visibility from the Nurse Station to all the Patient Rooms. The concept of visibility is not exactly well defined, it varies from a simple line of sight, to the idea of visibility and within reach, or even to the idea of visible from the circulation space [52, 53]. It has led to various design ideas related to the placement of nurse stations explored in this course [152]. Since it is not in the scope of this research to delve into the domain specific topic on this subject, the proof-of-concept takes one of the several possible approaches to this issue. The simple approach for the visibility selected in this example is a visibility rule that can be satisfied with constructing a line from the Nurse Station to the Patient Room's opening (Figure 96). The opening can be from a door or a window.

- The first question may be asked: how to determine the start and end point of the line of sight that will be acceptable as a measure the Patient Room is "visible" from the Nurse Station? The answer in this exercise is: A line drawn from center of the Nurse Station facing the Patient Room, to a center of an Opening to the Patient Room. This approach can be modified easily to connect a nurse station

represented by a desk (instead of a space), or the visibility of the patient bed instead of just an opening. These are possible as long as the model contains the relevant information.

- What can constrain the pairing of the Nurse Stations and the Patient Rooms since there are multiple Nurse Stations and Patient Rooms in the whole building? There are several ways to look at this. One simplest answer is simply perform a brute force approach by collecting all the Nurse Stations in one set and all the Patient Room in another. This approach will require computation for $(n \times m)$ combinatorial possibilities, plus more expensive computations to identify the valid line of sight. Instead of using the brute force, additional constraint can be introduced to filter the object sets into less combination pairs. There are three considerations that are takes: 1) it is reasonable to assume that the visibility rule also implies that a Patient Room is "served" by at least one Nurse Station that has the same circulation space; 2) each Nurse Station serves multiple Patient Rooms; and 3) There may be more than one common circulation space that is connected to a Nurse Station (Figure 96). In this exercise, it is decided that choosing a common space to "pair" the Nurse Station and Patient Rooms.
- How to determine a position to connect from the Nurse Station to the Opening at the Patient Room? This seemingly simple question requires more analysis to determine how to choose reasonable points. In this case the start point should be selected from the center of a boundary face between the Nurse Station and the common space (=Corridor), and end point to be the center of a face of an opening facing the same common space.
- Since the line of sight only ensures a point to point connection (without obstruction) from the Nurse Station to a Patient Room, it does not really tell whether the room itself is really visible and if it is how much is the coverage. To add to the line of sight analysis, an additional analysis is evaluated to determine

how much volume that is actually visible compared to the volume of the room. This follows the idea of a camera view frustum approach, which starts at the Nurse Station Location and through the room's opening viewable from that point of view.

8.2.1 The BIMRL statement for the hospital visibility rule

The BIMRL statement in the form of BIMRL triplets for the above visibility rule is described in the following sections.

8.2.1.1 The CHECK statement

The CHECK statement contains two Sets, names SET1 and SET2.

- SET1 is a query to select all IfcSpace of type Nurse Station, paired with the boundary information of a specific type, which is an IfcSpace of type Corridor.
- $\phi_1 = \{\phi_{(N,C1)} | \, N, C1 \in \phi_{IfcSpace} \wedge T(N) =' NurseStation' \wedge T(C1) = 'Corridor' \wedge (\phi_{BoundaryInfo(N)} \cap \phi_{BoundaryInfo(C1)}))$
- SET2 is a query to select all IfcSpace of type Patient Room, together with the boundary information of the room, which is of type IfcOpeningElement. The opening is in turn filtered only for those facing (or being a boundary of) a Corridor Space and is an opening for a door or a window only.
- $\phi_2 = \{\phi_{(P,C1)} | \, P, C1 \in \phi_{IfcSpace} \wedge T(P) =' Patient\ Room' \wedge T(C1) = 'Corridor' \wedge (\phi_{BoundaryInfo(P,D)} \cap \phi_{BoundaryInfo(C1,D)})\}; where\ D\ is\ an\ IfcOpeningElement$

```
CHECK
{ IfcSpace N, IfcSpace C1
   WHERE UPPER(N.longName) like '%NURSE STATION%'
      AND UPPER(c1.LongName) like '%CORRIDOR%'
   COLLECT N.ElementId NURSESTNEID, N.Name NSTNAME,
      N.LongName NURSESTNLNAME, C1.ElementId CORREID, C1.Name CORRNAME,
```

```
      C1.LongName CORRLNAME, BOUNDARYINFO(N,C1).CommonPointAtS BP1,
      BOUNDARYINFO(N,C1).SFacePolygon SET1FACE1,
      BOUNDARYINFO(N,C1).BFacePolygon SET1FACE2
} AS SET1;
{ IfcSpace P, IfcOpeningElement D, IfcSpace C2
   WHERE UPPER(P.LongName) like '%PATIENT ROOM%'
      AND BOUNDARYINFO(P,D).BoundaryElementId =
         BOUNDARYINFO(C2,D).BoundaryElementId
      AND UPPER(C2.LongName) like '%CORRIDOR%'
      AND C2.ElementType='IFCSPACE'
   COLLECT P.ElementId PATROOMEID, P.Name PATROOMNO,
      P.LongName PATROOMNAME, D.ElementId OPENINGEID,
      D.Name DOOROPENINGNAME, C2.ElementId CORREID,
      C2.LongName CORRLONGNAME, BOUNDARYINFO(C2,D).CommonPointAtS BP2,
      BOUNDARYINFO(C2,D).SFacePolygon SET2FACE1,
      BOUNDARYINFO(C2,D).BFacePolygon SET2FACE2,
      dependency(D,"WHERE DependentElementType in
         ('IFCDOOR','IFCWINDOW')").ElementId DOOREID
      } AS SET2;
```

8.2.1.2 The EVALUATE statement

The EVALUATE statement consists of two independent sub-clauses that executes the two evaluation functions: ComputeIntersection and VolumeIntersection. There are three distinct sets in operation in the first clause that uses ComputeIntersection.

- Input source for ComputeIntersection() that is implicit. It is formed by the JOIN operator from SET1 and SET2 from the CHECK statement: $\phi_{LineOfSight} = \phi_1 \cap \phi_2$ using CORREID as the join key. This join set is also the source for the geometry construction in CONSTRUCT clause that generates a LINE connecting BP1 (center of the space boundary face between the Nurse Station and the common Corridor) and BP2 (center of the room opening that faces the same common Corridor).

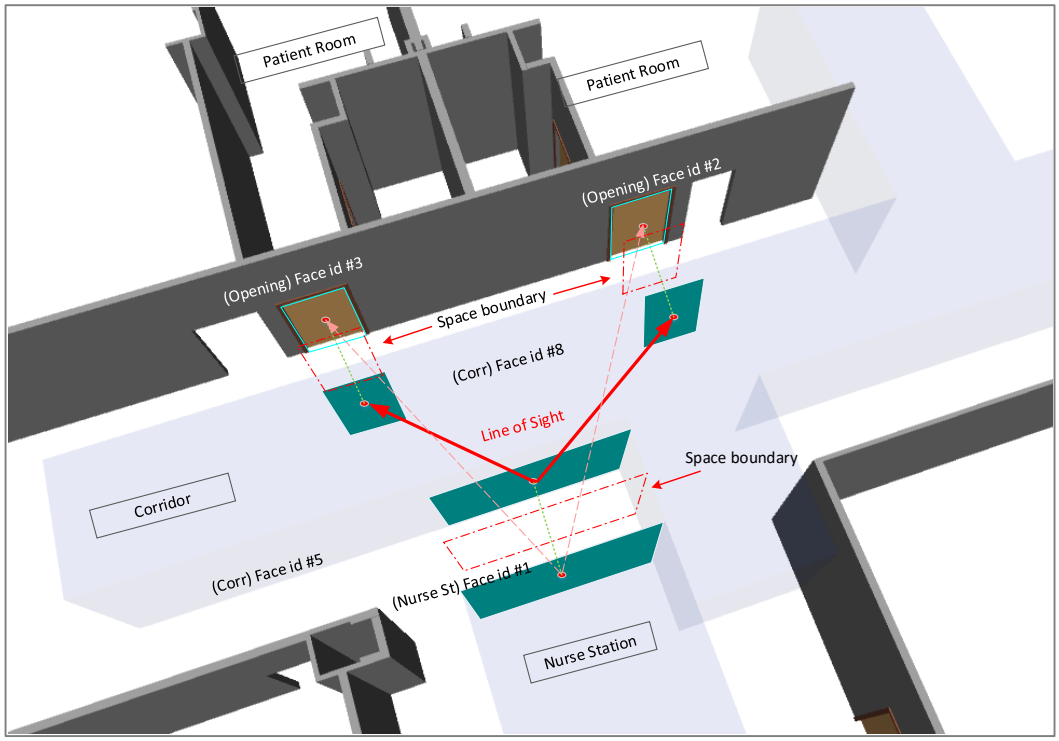


Figure 97 - Creation of Line of Sight Geometry

- Geometry set constructed by the Construct above. It is a set that connect pairs of the Nurse Station and Patient Rooms that share the same common corridor.
- A generic set of IfcElement that is supplied as a parameter to compute the intersection of the Line and any other element that are subtypes of IfcElement in the model.

The second clause in the EVALUATE statement uses two distinct sets:

- The first set $\phi_{SightView}$ is derived from the same source as the $\phi_{LineOfSight}$
- The Brep geometry set generated from $\phi_{LineOfSight}$ using a point at the center of the Nurse Station and the common corridor (the same starting point as the Line) and ends with the opening of the patient rooms that face the same corridor. This end face is extended by 20’ to compute the extended frustum that intersects with the patient room.

```
EVALUATE
{ COMPUTEINTERSECTION(LS, IfcElement, "EXCLUDEID NURSESTNEID, CORREID,
      PATROOMEID,OPENINGEID,DOOREID", EXACT)
      OUTPUT ?LineOfSight FOREACH GROUP OF (CORREID) FROM SET1
      JOIN SET2 USING (CORREID)
   CONSTRUCT LS (LINE (BP1, BP2))
};
{ VOLUMEINTERSECTION(SV,PATROOMEID) OUTPUT ?SightView
   FOREACH GROUP OF (CORREID) FROM SET1 JOIN SET2 USING (CORREID)
   CONSTRUCT SV (BREP(StartEndFaces(DefPoint(BP1), DefFace(SET2FACE2)
      EXTEND 20)))
};
```

8.2.1.3 The ACTION statement

Five actions are defined in the ACTION clause. Two clauses for the visibility evaluation and three for the volume intersection. They are segregated using the OUTPUT result from the evaluation functions, which is implicitly created when a variable ?LineOfsight is defined in the EVALUATE function.

- ?LineOfSight = 0. This is for the results when the visibility test returns NO IfcElement that blocks the line of sight. This clause performs two actions: Print and Draw. The Print action prints selected columns and save them into table LineOfSight. The draw action draws the Line created by Construct in the EVALUATE statement in Green. It is combined with the background model for IfcElement and IfcSpace objects from the architectural model (ModelId=2) and only on Level 2 and 3 (using the container ids). It excludes IfcCovering for improved visibility of the report and highlights the spaces that belong to the Nurse Stations and Patient Rooms using the specified color in RGB. The background model is set to 50% transparency for better visibility and is saved into an X3D file. These two actions are repeated almost identically for the other clauses. They

vary only in specific settings such as color, highlighted objects and the destination file.

- ?LineOfSight =1. It is the status for a line of sight that meets an object in its path.
- $0 \leq SightView \leq 0.05$. It collects all the frustum views that have only 5% or less coverage of the Patient Room.
- $0.05 \leq SightView \leq 0.15$. It collects all the frustum views that has between 5% or and 15% coverage of the Patient Room.
- $SightView > 0.15$. It collects all the frustum views that has a good 15% visibility into the patent room.

```
ACTION
   WHEN ?LineOfSight = 0 { PRINT NURSESTNEID, NURSESTNLNAME,
      PATROOMEID, PATROOMNO, PATROOMNAME, DOOREID, DOOROPENINGNAME,
      CORRLNAME SAVE into TABLE LineOfSight DRAW Color GREEN with
         BACKGROUND(IfcElement, IfcSpace) WHERE ModelId=2 AND
      Container in ('0ZsG1ZrQrCD87lTVoSuTLK','0ZsG1ZrQrCD87lTVoSuTLJ')
      AND ElementType!='IFCCOVERING' HIGHLIGHT (NURSESTNEID,
         PATROOMEID) Color RGB(141,74,230) TRANSPARENCY 0.5
      SAVE into X3D 'c:\temp\los0.x3d'
   };
   WHEN ?LineOfSight = 1 {
      PRINT NURSESTNEID, NURSESTNLNAME,
         PATROOMEID, PATROOMNO, PATROOMNAME, DOOREID, DOOROPENINGNAME,
         CORRLNAME SAVE into table LineOfSight APPEND
      DRAW Color magenta
         with BACKGROUND IfcElement WHERE (ModelId=2 AND container in
         ('0ZsG1ZrQrCD87lTVoSuTLK','0ZsG1ZrQrCD87lTVoSuTLJ') AND
         ElementType!='IFCCOVERING') or ElementId in
         (Select OutputDetails From UserGeom_OutputDetails)
         highlight (NURSESTNEID, PATROOMEID, OUTPUTDETAILS) Color CYAN
         TRANSPARENCY 0.5 SAVE into X3D 'c:\temp\los1.x3d'
   };
   WHEN ?SightView BETWEEN 0 AND 0.05 {
      PRINT NURSESTNEID,
```

```
      NURSESTNLNAME, PATROOMEID, PATROOMNO, PATROOMNAME, DOOREID,
      DOOROPENINGNAME, CORRLNAME SAVE into TABLE SightView
   DRAW Color RGB (233,152,71) with BACKGROUND (IFCELEMENT,
      IFCSPACE) WHERE ModelId=2 AND container in
      ('0ZsG1ZrQrCD87lTVoSuTLK','0ZsG1ZrQrCD87lTVoSuTLJ')
      AND ElementType!='IFCCOVERING' HIGHLIGHT (NURSESTNEID,
      PATROOMEID) Color cyan TRANSPARENCY 0.5 save into x3d
      'c:\temp\sv0.x3d' };
WHEN ?SightView BETWEEN 0.05 AND 0.15 {
   PRINT NURSESTNEID, NURSESTNLNAME, PATROOMEID, PATROOMNO,
      PATROOMNAME, DOOREID, DOOROPENINGNAME, CORRLNAME
      SAVE into TABLE SightView APPEND
   DRAW Color RGB (220, 236, 68) with BACKGROUND (IFCELEMENT,
      IFCSPACE) WHERE ModelId=2 AND container in
      ('0ZsG1ZrQrCD87lTVoSuTLK','0ZsG1ZrQrCD87lTVoSuTLJ')
      AND ElementType!='IFCCOVERING' HIGHLIGHT (NURSESTNEID,
      PATROOMEID) Color cyan TRANSPARENCY 0.5 SAVE into X3D
      'c:\temp\sv1.x3d' };
WHEN ?SightView > 0.15 {
   PRINT NURSESTNEID, NURSESTNLNAME, PATROOMEID, PATROOMNO,
      PATROOMNAME, DOOREID, DOOROPENINGNAME, CORRLNAME
      SAVE into TABLE SightView APPEND
   DRAW Color GREEN with BACKGROUND (IFCELEMENT, IFCSPACE)
      WHERE ModelId=2 AND container in
      ('0ZsG1ZrQrCD87lTVoSuTLK','0ZsG1ZrQrCD87lTVoSuTLJ')
      AND ElementType!='IFCCOVERING' SAVE into X3D
      'c:\temp\sv2.x3d' };
```

8.2.2 Aggregating multiple results

The above rule check does not exactly filter out the possibility of multiple connections, i.e. possibility that two spaces have more than one space boundary in between. The check above evaluates every possible connection and performs local checks as to whether the visibility requirement is met. The local check will only return a simple Boolean of true or false with regards to its local test. This means a pair of spaces (a Nurse Station and a Patient Room) that have more than one possible visibility path may pass in

one and fail in another. To aggregate such results, a simple query to aggregate the results can be done thanks to the integration of BIMRL with the RDBMS. The following query will perform the aggregation:

```
SQL select nursestneid, patroomeid, patroomno, patroomname
      from lineofsight having sum(output)>0 group by nursestneid,
         patroomeid, patroomno, patroomname
   minus
    select nursestneid, patroomeid, patroomno, patroomname
      from lineofsight where output=0;;
```

The statement collect all results of the pair ids of a Nurse Station and a Patient Room that has a positive result (sum(output) > 0) and subtract it with those that has at least one negative result (output=0). This query will return only pairs that return fail for all the possible visibility paths. Every case that has at least one solution that returns success will be excluded.

8.2.3 Checking for full coverage

The above check works with spaces that can be found with proper relationships that meet the following conditions:

1. The spaces concerned, i.e. Nurse Stations and Patient Rooms are connected through a common circulation space
2. All the spaces are created correctly
3. The assumption of access through a common space or corridor is universally true

Conditions 1 and 3 are explicitly built into the BIMRL rule above. Condition 2 is hard to check since the non-existence of an object in a model should be checked by a different rule that maintains data quality to be done in the authoring tool, for example by simply comparing the space schedule with the expected number of spaces to be created in the model (from an external source). Conditions 1 and 3 can be verified with another rule subsequent to the above BIMRL rule:

```
CHECK
   IfcSpace S where longname like '%PATIENT ROOM%' AND
      Elementid not in (select patroomeid from lineofsight)
   COLLECT elementid patroomeid, name patroomno,
      longname patroomname;
EVALUATE
   NOTHING() OUTPUT ?inAcc;
ACTION
   WHEN ?inAcc {
      PRINT patroomeid, patroomno, patroomname
      DRAW COLOR Magenta with 0 Transparent
         background (IfcElement, IfcSpace)
         WHERE MODELID=2 AND container='0ZsG1ZrQrCD87lTVoSuTLK'
         AND ElementType!='IFCCOVERING' HIGHLIGHT (PATROOMEID)
         COLOR Red TRANSPARENCY 0.5 SAVE INTO X3D
         'c:\temp\losInacc1.x3d'
   };
   WHEN ?inAcc {
      DRAW COLOR Magenta with 0 Transparent
         background (IfcElement, IfcSpace)
         WHERE MODELID=2 AND container='0ZsG1ZrQrCD87lTVoSuTLJ'
         AND ElementType!='IFCCOVERING' HIGHLIGHT (PATROOMEID)
         COLOR Red TRANSPARENCY 0.5 SAVE INTO X3D
         'c:\temp\losInacc2.x3d'
   };
```

8.3 Case-2: Singapore's Environment and Safety Rule

(42) 3.2.2 Design Criteria

f) The discharge pipe shall not be located in places where it can cause health and safety hazards such as locating the discharge pipe above any portable water storage tank and electrical transformer/ switchgear.

The above rule is taken from a scope in the CORENET IBS project. This rule is part of the code of practice for environmental and health related topics. It specifies a requirement for the positioning of a drinkable water tank and electrical transformer or switchgear. The main consideration for this requirement is for the health and safety of drinkable water from any possible contamination and possibility of fire due to the waste water leakage from the pipe. The requirement is important considering the case of a contamination in Singapore by exactly the same reason that occurred just a few years earlier in 2000 [153].

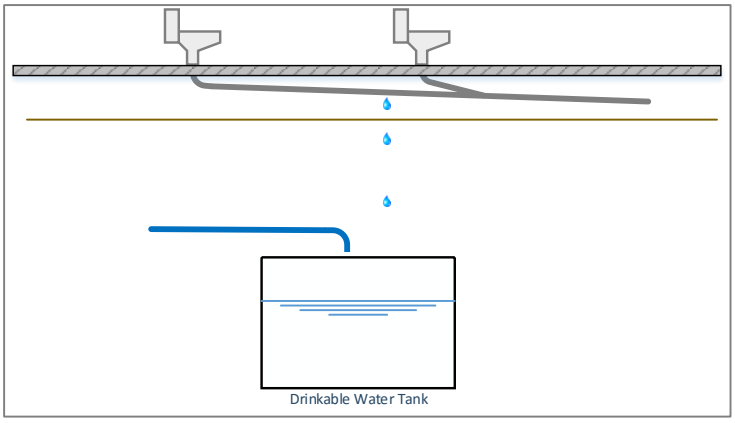


Figure 98 - The Danger of Contamination due to Leakage from a Waste Water System

The CG for this rule has been described in Figure 27 of Chapter 4.3.1.2. There are two sets of objects that are involved in this rule: Pipe segments that belong to a sanitary system and potable water tank, and a switchgear or transformer. The first impression from the reading of the rule is that the main object to be checked is the discharge pipe. In this exercise however the inverse is chosen, i.e. the set of Tank, Transformer or Switchgear as the main object (Figure 99). The main motivation for this approach is for practical reasons. In a building model with complete MEP systems, each of the system usually has a large number of members. In this case if the discharge pipe is selected as the main object, the candidate set will be much larger than if the other set of objects are selected, which require a lot more computation to evaluate every single member for compliance to this rule. On the other hand, there are not many Storage Tanks,

Transformers or Switchgears in the entire building. Therefore, it will be computationally more efficient to evaluate the requirement based on the second set of objects.

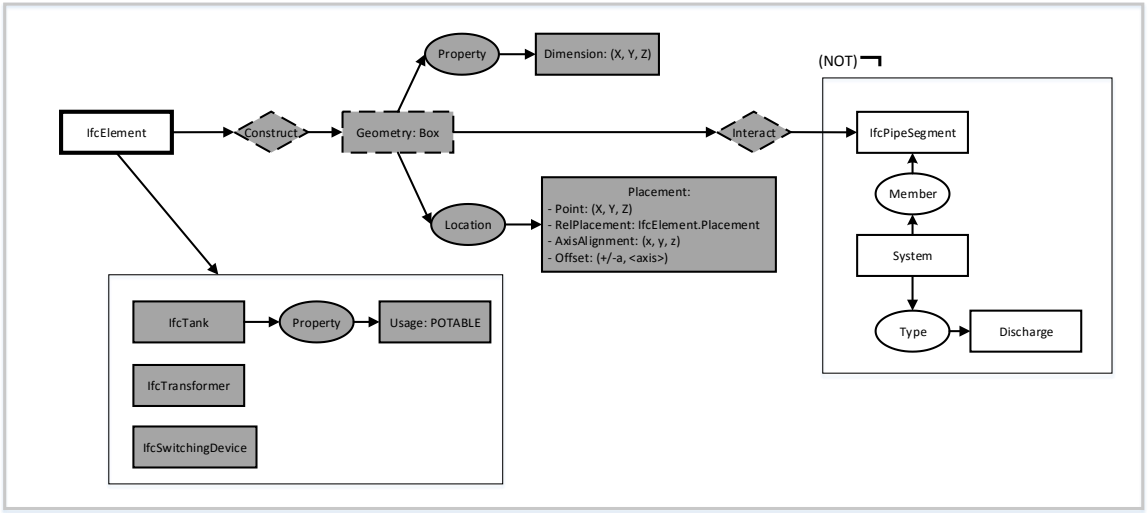


Figure 99 - Inverted CG for Case-2 rule

8.3.1.1 The CHECK statement

Two initial sets are defined for the CHECK statement:

1. Set 1 ϕ_1: Set of IfcSwitchingDeviceType, IfcTransfomerType, or IfcTankType with special property 'POTABLE' in ObjectType attribute.
2. Set 2 ϕ_2: Set of the distribution elements that belong to a sanitary system.

Note that the OBB geometry (Chapter 6.2.3) is being used in this rule. The OBB is the most suitable geometry type to represent the shape of the objects as they are generally irregular in shape. The projected OBB is selected since it will give much better bounding box information for the object independent of their placement relative to the axes (Figure 100).

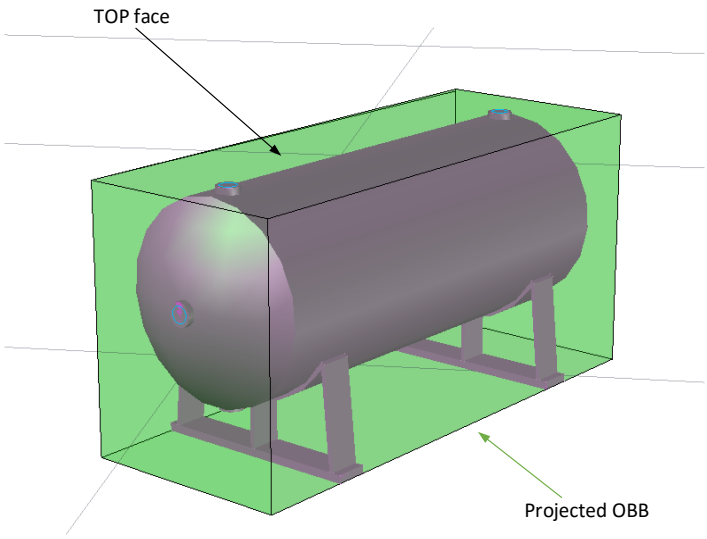


Figure 100 - Projected OBB on a Horizontal Water Tank

CHECK

```
{ IfcElement E, IfcBuildingStorey Y
   WHERE (TYPEOF(E).ifctype in
      ('IFCSWITCHINGDEVICETYPE','IFCTRANSFORMERTYPE') OR
      (TYPEOF(E).ifctype='IFCTANKTYPE' AND e.objecttype like
         'POTABLE%'))
      AND TOP(E).type='PROJOBB' AND (CONTAINER(E, Y,
         "WHERE ParentType='IFCBUILDINGSTOREY'")=.T.)
   COLLECT e.elementid OBJEID, E.NAME NAME, TYPEOF(E).ifctype IFCTYPE,
      TYPEOF(E).name TYPENAME, TOP(E).POLYGON FACEGEOM,
      Y.Name STOREYNAME, PROPERTY(Y,ELEVATIONHEIGHT) STELEVATIONHT,
      TOP(E).Centroid.sdo_point.Z ELEMHEIGHT
} as SET1
{ IfcDistributionElement D
   WHERE SYSTEMOF(D).objecttype like 'Sanitary%'
   COLLECT D.ELEMENTID MEPOBJEID, D.name MEPOBJNAME,
      SYSTEMOF(D).name SYSTEMNAME
} as SET2;
```

8.3.1.2 The EVALUATE statement

This rule requires a check to see if there is any sanitary pipe located overhead that may be concealed by the ceiling. To perform this check, a generic ComputeIntersection()

function is used for an intersection between ϕ_2 and a geometry constructed from the top face of the projected OBB of $\phi_1$objects extruded vertically up. In this example, the extrusion height is calculated from the TOP face of an object to the height of the story. This is sufficient to cover the typical story height of a building to check for the existence of a member of a sanitary system.

There is an assumption made in this rule definition, i.e. the property that defines the elevation height (the upper height of a building story) is known and available. In this example, the property values are added into the model. The value is important to define the correct upper limit for the box extrusion.

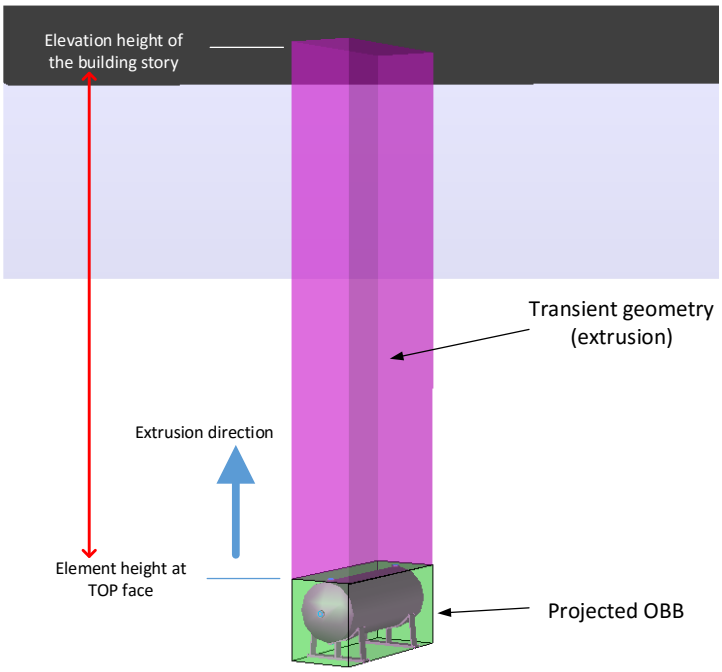


Figure 101 - Creation of Extruded Transient Geometry

```
EVALUATE
   COMPUTEINTERSECTION(EB, SET2.MEPOBJEID) OUTPUT ?clash FROM SET1
      CONSTRUCT EB (EXTRUSION(DefFace(FACEGEOM), +ZAXIS,
         STELEVATIONHT-ELEMHEIGHT));
```

8.3.1.3 The ACTION statement

The intersection results reported in the variable ?clash (OUTPUT column) contains 0 (no intersection) or 1 (with intersection). Two action statements are specified, one for the case without intersection (?clash = 0), and the other one with intersection or a non-compliance case (?clash > 0). For both, the evaluation results are aggregated into a single table named Contamination. Each of the action writes separate X3D files, mainly for the purpose of study. Using both modes (Print and Draw), one can analyze the building design and its responses to this rule check. The X3D report offers an interactive and visual insight into the building model and the rule checking behavior.

```
ACTION
   WHEN ?clash = 0 {
      PRINT RESULT SAVE INTO TABLE contamination
      DRAW COLOR CYAN TRANSPARENCY 0.5 WITH TRANSPARENT BACKGROUND
         (IfcDistributionElement, IfcSlab, IfcCovering)
         highlight (OBJEID, OUTPUTDETAILS) COLOR RED TRANSPARENCY 0
         SAVE INTO X3D 'c:\temp\NoContamination.x3d'
   };
   WHEN ?clash > 0 {
      PRINT RESULT SAVE INTO TABLE contamination APPEND
      DRAW COLOR MAGENTA TRANSPARENCY 0.5 WITH TRANSPARENT BACKGROUND
         (IfcDistributionElement, IfcSlab, IfcCovering)
         highlight (OBJEID, OUTPUTDETAILS) COLOR RED TRANSPARENCY 0
         SAVE INTO X3D 'c:\temp\contamination.x3d'
   };
```

8.4 Case-3: Number of exits from rooms and spaces

Singapore Fire Code 2013

Chapter 2 Means of Escape

2.2.10 Number of exits from rooms and spaces

There shall be at least two door openings remote from each other and leading to exits from every room or enclosed space in which the total occupant load exceeds the maximum permissible occupant load for one door as listed in the table below:

Type of Occupancy	Maximum Occupant Load with One Door
High Hazard	25
Patient accommodation area	50
Classrooms	50
Godowns, stores, and factories not being of high hazard type	50
Assembly	50
Rooms and spaces with occupancy of more than 50 persons shall comply with the requirements for `Number and Width of Exits' under Cl.2.8.2 for Assembly Occupancy.	
Note: i. For residential occupancy, see cl.2.4. ii. For health care occupancy, see cl.2.5. iii. For office/shop/factory/warehouse occupancy, see cl.2.6. iv. For hotels, see cl.2.7. v. For assembly occupancy, see cl.2.8.	

Note: Occupant load can also be determined using the following formula:

$$Occupant\ Load = \frac{Gross\ floor\ area}{Occupant\ load\ factor\ of\ 10m^2\ per\ person}$$

No of Occupants	Min No if Doors	Min. Width of Corridors
51 – 200	2	1200 mm
201 – 500	2	1250 mm
501 – 1000	3	1250 mm
Exceeding 1000	4	1250 mm

A similar rule to the one above can also be found in the IBC 2009 code. Here the criteria for the number of remotely located exits is measured by the area of the room [67].

> ***1014.2.3.2 Exit access.*** *Any patient sleeping room, or any suite that includes patient sleeping rooms, of more than 1,000 square feet (93 m2) shall have at least two exit access doors remotely located from each other.*
>
> ***1014.2.4.2 Exit access.*** *Any room or suite of rooms, other than patient sleeping rooms, of more than 2,500 square feet (232 m2) shall have at least two exit access doors remotely located from each other.*

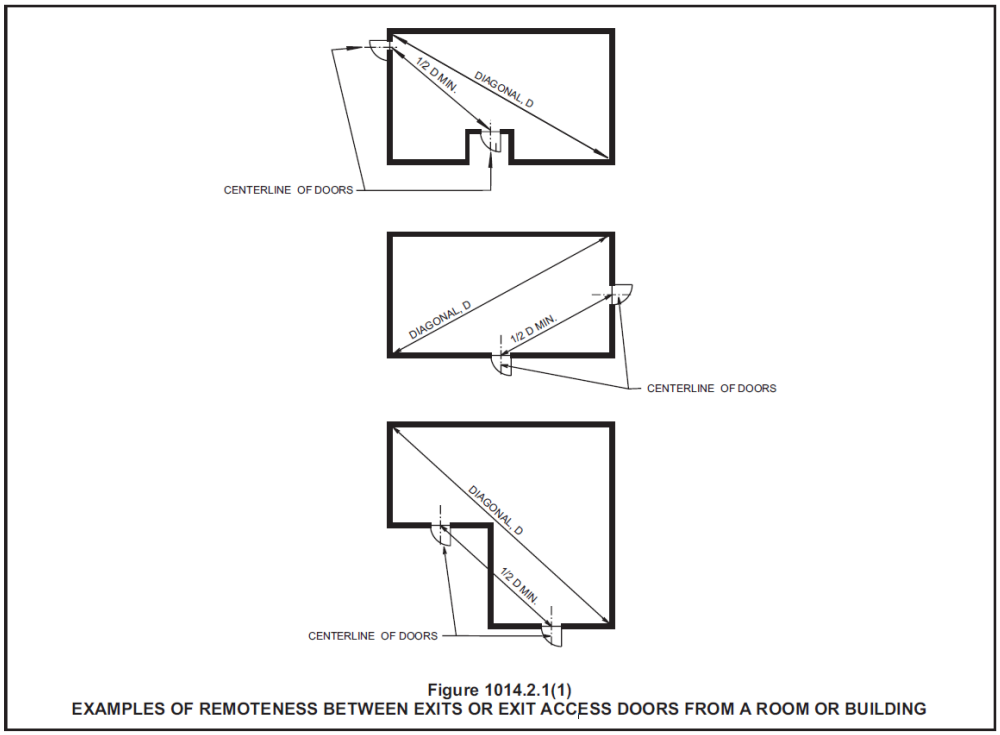


Figure 1014.2.1(1)
EXAMPLES OF REMOTENESS BETWEEN EXITS OR EXIT ACCESS DOORS FROM A ROOM OR BUILDING

Figure 102 - Explanation of Remoteness between 2 Exits [70]

In this test case, the focus will be on the rule to determine the remoteness of the exit doors. The basic rule formulation can be described as follows:

- Given a set of spaces S in the model ϕ_S, find all spaces that has occupancy number larger than 50 ϕ'_S.
- For each space in ϕ'_S, identify exit doors D_e
- Two measurements are needed: count the number of exit door (must be >= 2), and measure the distance ratio to the diagonal distance of the room: $r = \frac{D_{exit\_1,exit\_2}}{D_{s,diagonal}}$

The complexity arises when examining the details to be considered before a rule can be defined in BIMRL:

1. The most straightforward definition of space occupancy is based on IFC property set: Pset_SpaceOccupancyRequirements that has a property OccupancyNumber. This property set is expected to be assigned to a space. Other than this

information, is there any other way this information can be inferred from the model?

2. There are potentially many doors that are in a space, for example a door to a balcony, to a bathroom, etc. In this case only exit doors are relevant. One way to determine an exit door is by checking the property attached to the Door Pset_DoorCommon that has FireExit property. Another possibility is by looking into the name or classification, or a more intelligent way is to check that a door that is connected to a circulation space outside the room is an exit.
3. The measurement of distance needs to consider the space(s). In some cases, there may be more than one space involved. The diagonal needs to be measured from the combined spaces' OBB. This is compared to the distance between doors that are calculated measuring from the centroid to the centroid.

8.4.1 Sub-rule #1

In this example, this rule will be represented in two steps, i.e. two rules. The first rule collects qualified spaces for the following information:

- Gather all spaces that have occupancy number more than 50. It may come from various sources:
    - OccupancyNumber property from Pset_SpaceOccupancyRequirements
    - Area of the space (from its footprint). In this case it is calculated based on 10 m^2 per person, which is > 500 m^2 or (5,382 sq ft) for 50 occupants or more (Set1 together with the above criterion).
    - Count the seating capacity based on the number of seats in the rooms (Set2)
    - Count the number of Sprinklers inside the space for information to determine the distance between exits: $\frac{1}{2}D_{diagonal}$ for a non-sprinkler protected space or $\frac{1}{3}D_{diagonal}$ for a sprinkler protected space.

- In the EVALUATE statement, only join is used to merge the above information into one. NOTHING() evaluation function is used here as a dummy function that does nothing.

  In this example the use of both methods of rule chaining is demonstrated. In this first sub-rule, there are two evaluation functions. In the first evaluation function SET1 is joined to SET2 and the output will go to ?Occupancy internal variable that also represents an intermediate table. This table is then used in the second evaluate statement by selecting only the spaces that fulfil the following conditions: OccupancyNo>=50 or SeatingCapacity>=50, or SpaceArea>=5382, and joins it with SET3 to give the final evaluate result table SprinklProt.

- In the ACTION statement, it simply saves the final result into SpaceOccupancy table.

```
SETVAR ?occNo := 50;
SETVAR ?occNoArea := 5382;
SETVAR ?seating := '%Seating%';
CHECK
   { IfcSpace SA, BIMRL_TOPO_FACE F
      Where (PROPERTY(SA,OccupancyNumber,TO_NUMBER)>=?occNo or
         sdo_geom.sdo_area(F.polygon,0.000001)>=?occNoArea) and
         F.elementid=SA.elementid and F.Orientation='BOTTOM'
      Collect SA.elementid SPACEID, SA.name SPACENAME,
         SA.longname SPACELNAME, Property(SA,OccupancyNumber,
            TO_NUMBER) OCCUPANCYNO,
         CONTAINER(SA).name STOREYNAME,
         CONTAINER(SA).longname STOREYLNAME,
         sdo_geom.sdo_area(F.polygon,0.00001) SPACEAREA
   } as Set1;
   { IfcSpace SB
      Where CONTAINS(SB,USEGEOMETRY).ElementType=
          'IFCFURNISHINGELEMENT' and
          CONTAINS(SB,USEGEOMETRY).Name like ?seating
       COLLECT SB.ElementId SPACEID, SB.Name SPACENAME,
          SB.LongName SPACELNAME,
```

```
            COUNT(unique CO.ElementID) SEATINGCAPACITY
            GROUP BY SB.ELementId, SB.Name, SB.LongName
      } as Set2;
      { IfcSpace SC
         Where CONTAINS(SC,USEGEOMETRY).ElementType=
            'IFCFLOWTERMINAL' and CONTAINS(SC,USEGEOMETRY).Name
            like 'Sprinkler%'
         COLLECT SC.ElementId SPACEID, SC.Name SPACENAME,
            SC.LongName SPACELNAME,
            COUNT(unique CO.ElementID) SPRINKLERNO
            GROUP BY SC.ELementId, SC.Name, SC.LongName
      } as Set3;
EVALUATE
      { NOTHING() Output ?Occupancy
         From SET1 FULL OUTER JOIN SET2 USING (SPACEID, SPACENAME,
            SPACELNAME)
      };
      { NOTHING() Output ?SprinklProt
         From (select * from Occupancy where occupancyno>=?occNo or
            seatingcapacity>=?occNo or spacearea>=?occNoArea)
            LEFT JOIN SET3 USING (SPACEID, SPACENAME, SPACELNAME)
      };
ACTION
      WHEN ?SprinklProt {
         PRINT RESULT SAVE INTO TABLE SpaceOccupancy };
```

8.4.2 Sub-rule #2

The second sub-rule demonstrated the second mode of rule chaining by defining a rule that uses the result from the previous sub-rule.

8.4.2.1 The CHECK statement

In the CHECK statement, the first set (Set1) uses a SpaceOccupancy table that stores the result from the first sub-rule, which keeps the information of the spaces that has occupancy number above 50. The footprint of each of the space (the bottom face) will be used to compute the diagonal distance of the OBB in the evaluate function. The second

CHECK statement gathers the exit doors that is defined to be all the doors that lead into a circulation space, which has an OmniClass classification code of 13-25 *nn nn*, where *nn* can be of any number that specifies the detailed type of the circulation space category (13-25 00 00). The third set (Set3) is a collection of all doors that have the property FireExit.

```
CHECK
{ IfcSpace S
   WHERE S.Elementid in (select unique SpaceId
      from SpaceOccupancy)
   COLLECT S.elementid SPACEID, S.name SPACENAME,
      S.longname SPACELNAME, BOTTOM(S).polygon FOOTPRINT,
      CONTAINER(S).name STOREYNAME,
      CONTAINER(S).longname STOREYLNAME
} as Set1;
{ IfcSpace A, IfcOpeningElement D, IfcSpace C
   WHERE BOUNDARYINFO(A,D).BoundaryElementId =
      BOUNDARYINFO(C,D).BoundaryElementId
      AND CLASSIFICATIONOF(C).ClassificationItemCode like
      '13-25 __ __' AND
      DEPENDENTTO(D).ElementType='IFCWALLSTANDARDCASE'
   COLLECT D.ElementId DOORID, A.Elementid SPACEID,
      A.NAME SPACENAME, A.LONGNAME SPACELNAME, D.NAME DOORNAME,
      D.ElementId OPENINGID, D.Body_Major_Axis_Centroid
         OPENINGCENTROID, C.Name CIRCSPACE
} as Set2;
{ IfcDoor ED
   WHERE PROPERTY(ED,FireExit)='true'
   COLLECT BOUNDEDSPACE(ED).ElementId SPACEID,
      BOUNDEDSPACE(ED).Name SPACENAME,
      BOUNDEDSPACE(ED).LongName SPACELNAME,
      ED.ElementId OPENINGID,
      ED.Body_Major_Axis_Centroid DOORCENTROID
} as Set3;
```

8.4.2.2 The EVALUATE statement

In the evaluate statement, ComputeRemoteLocation() function is used. It accepts five parameters or more.

- The first parameter is for the main object id for this check. This should be a building story, or a space (or the main space in focus that represents a collection).
- The second parameter is detailed information associated with the object in the first parameter, usually in a collection. In this example, since the rule is to check an individual space, both parameters 1 and 2 use the same space ids.
- The third parameter is the footprint geometry (bottom face in most cases) of the individual object(s) specified in the parameter 2. This information is used to compute the OBB.

  The first three parameters are used together to compute the OBB. The detailed objects in parameter 2 will be grouped into one set based on their shared id of the first parameter. Their associated footprints will be collected to calculate OBB of the collection. This is particularly useful if the main object in parameter 1 does not have its own geometry, e.g. building story, or a collection of spaces.
- Parameter 4 contains the id of the second set of information. In this case it contains the doors that is collected in the Set2 that will provide the information to calculate the direct distance between the exits. The number of distinct object ids in this set grouped by the first parameter determines how many exits the main object has.
- Parameter 5 and above (if any) is the geometry, i.e. a location that represents each of the objects in parameter 4. The parameters allow for a different column containing the geometry from the collection of objects in parameter 4 to be aggregated. If there is more than one value, only the first value will be used. For multiple doors in a space, the function will calculate a direct distance between all

the possible pairs and compare them with the diagonal distance of the OBB calculated from the grouped information from parameter 3. The keyword "FOREACH GROUP OF AGGREGATE" information informs the function that the information that follows it is to be aggregated.

```
EVALUATE
   { ComputeRemoteLocation(SPACEID,SPACEID,FOOTPRINT,
         OPENINGID,OPENINGCENTROID,DOORCENTROID) output ?ratio
      foreach group of AGGREGATE (SPACEID, SPACENAME, SPACELNAME)
      from Set1 join Set2 using (SPACEID, SPACENAME,
         SPACELNAME)
   };
```

8.4.2.3 The ACTION statement

The ACTION statement defines two actions, one for the cases when the distance ratio fulfils the requirement, i.e. larger the pass ratio prescribed. The other one is for the failed cases. Both results are merged into one table named remotelyLoc but exported separately into two different X3D files with different colours.

```
ACTION
   WHEN ?ratio < ?passRatio {
      PRINT RESULT SAVE INTO TABLE remotelyLoc
      DRAW COLOR RED WITH TRANSPARENT BACKGROUND
         (IfcBuildingElement)
      SAVE INTO X3D 'c:\temp\remotelyLocF.x3d'
   };
   WHEN ?ratio >= ?passRatio {
      PRINT RESULT SAVE INTO TABLE remotelyLoc APPEND
      DRAW COLOR GREEN WITH TRANSPARENT BACKGROUND
         (IfcBuildingElement)
      SAVE INTO X3D 'c:\temp\remotelyLocP.x3d'
   };
```

8.5 Case-4: Accessibility Path Analysis using Circulation Graph

There are many rules that need to make use of graph information, such as egress requirements, MEP system tracing, accessibility requirements, etc. In this proof-of-concept, a simple circulation graph is created and using an evaluation function, a dynamic path analysis or query can be achieved easily. The test case involves a simple requirement to ensure that every patient room in Model-E is accessible from the main entrance represented by a Vestibule space as a starting point. Access to second and third level is possible through 2 sets of elevators and 2 staircases (Figure 103. The figure only shows one set of each that are nearest to the main entrance).

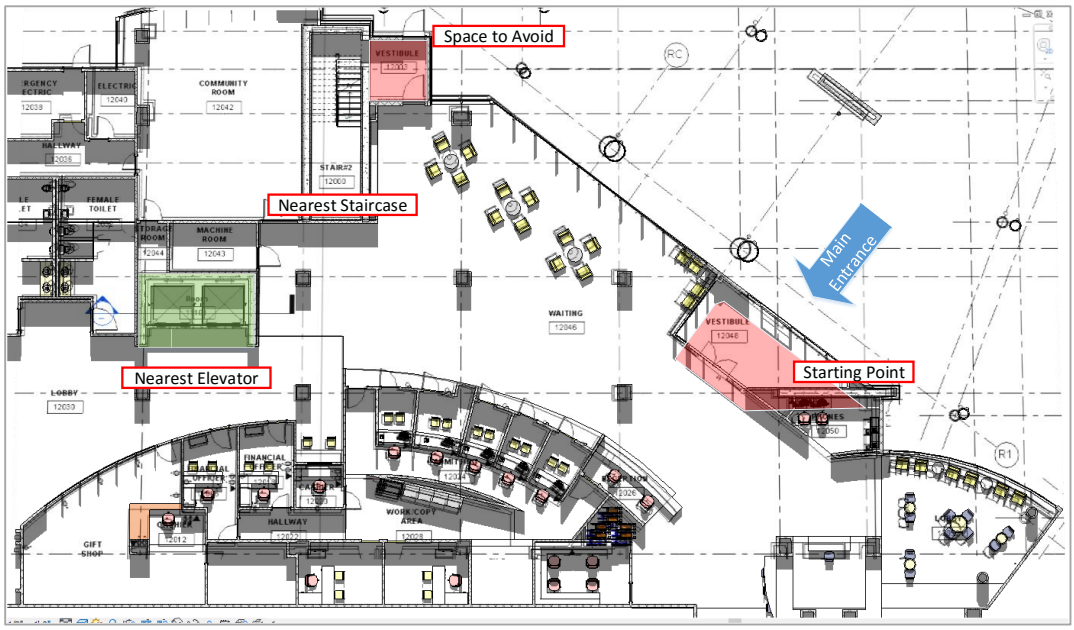


Figure 103 - Plan View of the Entrance Vestibule as the Starting Point for Accessibility Path

The BIMRL triplet statement in its simplest form of an accessibility rule is shown below:

```
CHECK
   IfcSpace S, IfcSpace E
      WHERE upper(S.LongName) like 'VESTIBULE%' and
         upper(E.LongName) like '%PATIENT ROOM%'
      COLLECT S.ElementID StartID, S.Name StartName,
```

```
        S.LongName StartLName, E.ElementID DestID,
        E.Name DestName, E.LongName DestLName;
EVALUATE
   ComputePath(StartID, DestID, "AVOID StartElemName='12003'
      or EndElemName='12003'") Output ?PathFound;
ACTION
   When ?PathFound = 1
      { PRINT RESULT SAVE INTO TABLE CIRCPATH DRAW COLOR RED
          With BACKGROUND (IfcSpace) HighLight (startID, DestID)
          COLOR CYAN TRANSPARENCY 0.75
          Save Into X3D 'c:\temp\pathFound.x3d'
      };
   When ?PathFound = 0
      { PRINT RESULT SAVE INTO TABLE CIRCPATH APPEND DRAW
         with BACKGROUND (IfcSpace) HighLight (startID, DestID)
            COLOR MAGENTA
            Save Into X3D 'c:\temp\pathNotFound.X3D'
      };
```

This rule searches every pair of Vestibules and Patient Rooms in the CHECK query. ComputePath() is the evaluation function using an extension method that is responsible for computing the shortest path for each pair. The function implements additional keywords that can control the graph generation and the shortest path decision. The two keywords are:

1. EXCLUDE. Exclude specifies that objects that meet the specified criteria will be de-activated, i.e. will not be included in the graph. This keyword ensures that the path will never be available for the path computation. This is a very useful option that can be used in various scenarios, for example to compute an egress path where no elevator should be used at all. In this case, all links involving elevators must be disabled.
2. AVOID. Avoid uses a different mechanism. It basically increases the cost for using the specified path significantly. The Avoid keyword will cause the path computation to put the option going through the specified link(s) as a lower

priority, but will not eliminate it completely. This means that if there is no other alternative, this path will still be chosen.

In this example, the rule specifies that a vestibule space, which leads to a staircase (room number 12003) is to be avoided. This forces the computation to choose other alternatives first.

Other variations of the rule may use the following keywords:

1. Exclude all elevators and only uses staircases, and avoid passing through kitchen spaces, simulating egress requirements (IBC 2009 1014.2 (4) Egress through intervening spaces) [67].

```
EVALUATE ComputePath(StartID, DestID, "EXCLUDE link_type like
   'VERTICAL CIRCULATION BY ELEVATOR'", "AVOID
   Upper(StartElemLName) like '%KITCHEN%' or
   Upper(EndElemLName) like '%KITCHEN%'" ) Output ?PathFound;
```

2. Avoid the use of Stairs, so that elevators are preferred.

```
EVALUATE ComputePath(StartID, DestID, "AVOID link_type like
   'VERTICAL CIRCULATION BY IFCSTAIR%'") Output ?PathFound;
```

3. Avoid the use of spaces that belong to different security zones as an example in the GSA courthouse requirements.

```
EVALUATE ComputePath(StartID, DestID, "AVOID Start_node_name in
   (select MemberElementId from BIMRL_RelGroup_0031 a,
      BIMRL_Element_0031 b where a.GroupElementId=b.ElementId
      and b.El ementType='IFCZONE' and
      b.ObjectType='SECURITY ZONE 3')") Output ?PathFound;
```

There are many more possibilities using graphs in combination with database queries that are shown in this test case. This test case demonstrates the use of integrated graphs in the database and its potential uses for dynamic path analysis using just one evaluation function. For more specialized requirements, additional extension functions may be developed.

8.6 A Summary of the Coverage of the Proof-of-Concept Test Cases

The four test cases above are chosen to give a good representation of the coverage of the rule checking problems. In general there are two measures that one can use to define the breadth of coverage of rule checking problems. The first measure is based on the high level classifications of rules as described in CHAPTER 3 regarding the extended concept that can be supported and the capability to support geometry, spatial and graph operations. The second measure is a more detailed one based on types of checking. This classification does not yet exist and in the current context can only be measured from practical experience developing automated rule checking systems. The summary of the coverage of the four test cases is shown in Table 16.

Table 16 - Summary of the Coverage of Rule Checking Problems for the Four Proof-of-Concept Test Cases Used in this Validation

Test case #	Coverage	Remarks
#1 Hospital Design – Best Practice	- Advanced filtering and association between objects, i.e. nurse stations and patient rooms - Uses geometry and spatial operations - Extended concept: visibility (using line of sight and sight view percentage) - Dynamically created geometry - Support multiple independent evaluation functions on the same set of objects	- Most rules require filter conditions that vary from one to another greatly. A query system was developed in CORENET ePlanCheck to query IFC data inside an EDM model server. - Large percentage of complex rule requirements in CORENET ePlanCheck require geometry, spatial operations, and dynamically created geometry - The concept of visibility is used in several complex rules such as fire exit requirements - Rule requirements often includes specific computations that involves customization. In
#2 Singapore's Environment and Safety Rule	- Ability to use advanced filters using types and a higher level IFC object definition - Uses dynamically created geometry - Uses efficient spatial query to perform intersection checks in a large model	

#3 Number of exits from rooms and spaces	- Demonstrates ability to extend evaluation function - Ability to perform geometry related queries for diagonal distance of a footprint of a group of spaces, and distance calculation - Use both rule chaining and evaluation function chaining supporting intermediate state.	all existing known implementations, it requires significant effort
#4 Accessibility Path Analysis using Circulation Graph	- Support of graph inside the database - Dynamic query to the graph - Ability to alter the graph dynamically for different desired analysis results	- Many advanced rules require graphs, for examples for circulations analysis, and large number of rules in the MEP domain - Dynamically altered graph behaviour is currently an unknown feature supported in any known automated rule checking implementation

Based on the above summary, the various aspects of capability used in the four test cases, including the ability to extend the evaluation function using a plugin mechanism that encapsulates complex checking logic or very specific checking requirements, provides a wide coverage of the fundamental features that rule checking systems such as CORENET ePlanCheck are built upon. Many of the features are not available in Solibri Model Checker. The rule chaining mechanism also offers significant flexibility that allows intermediate states to be captured for complex rules or set of related rules.

It is envisaged that the basic or standard evaluation functions will grow over time and will make rule definition easier. In addition, further improvements to support more types of transient geometry may be added to give more sophisticated possibilities to create different types often used in code checking, for example segmentation of edges that is useful to evaluate sections of an object for a localized test in various requirements such as disabled access requirements or protection from falling. Adding an actual path

between two points that extends a direct line of sight for an access path also will benefit many egress path related requirements and various circulation related rules. Such functionalities are finer grain specialization functions that do not affect the fundamental coverage of BIMRL.

In summary, the four test cases selected in this proof-of-concept cover a significant breadth of rule checking requirements and checking type at the most fundamental level. Without BIMRL, the four test cases will require a significant amount of effort to build them using a programmatic approach even with existing platforms such as FORNAX, or Solibri.

CHAPTER 9

PROOF-OF-CONCEPT VALIDATION

To validate the effectiveness of the concepts that BIMRL uses to solve rule checking problems, the rules defined in the previous chapters are applied to actual building models. The outcome will be presented and discussed in this chapter.

9.1 Case #1 Test and Validation: Visibility Rule in Hospital Design

This test case requires a hospital model with the necessary information in it. The model must have Nurse Stations modeled as the space at minimum. It also requires the common circulation spaces to be modelled and identifiable. The model used to validate this rule is Model-E (APPENDIX B). It is a three-story building with two story wards. The building is segregated into West and East wings, each with a dedicated Nurse station to serve the patient rooms. This design uses centralized Nurse Stations for each wing. The floor plan for story-3 is shown in Figure 104. The story-2 floor plan is identical with the third story.

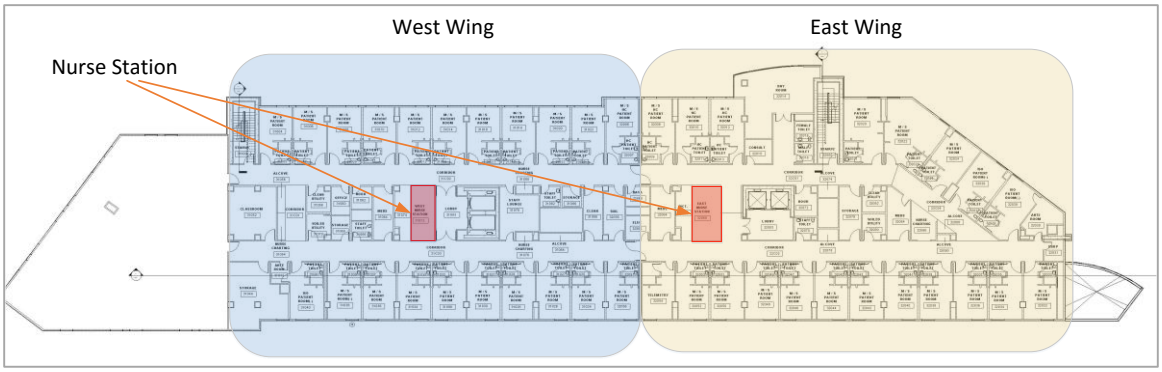


Figure 104 - Level-3 Floor Plan of the Hospital Model (Model-E)

9.1.1 Direct Line of Sight Check

Applying the BIMRL rule defined in the chapter 8 produces results showing direct line of sight from a Nurse Station to a Patient Room. Figure 105 shows the results of a direct line of sight check. Some lines of sight are blocked by a building element (value in ?LineOfSight variable = 1) seen in the Figure 105(B), whereas others pass (value in ?LineOfSight variable = 0) as seen in Figure 105(A) and (C). In the East Wings, there are two candidates for each pair of the Nurse Station and a Patient Room due to the shape of the Nurse Station and the Corridor that creates two faces that connect both (Figure 105(A)). In this case, any one line that passes the test is sufficient. Therefore an additional query is required to aggregate the data. Section 8.2.2 describes the details of this SQL statement. From the SQL query, it returns two rooms that do not have an unhindered line of sight from the nurse station, one at each level (Table 17 and Figure 105(B)).

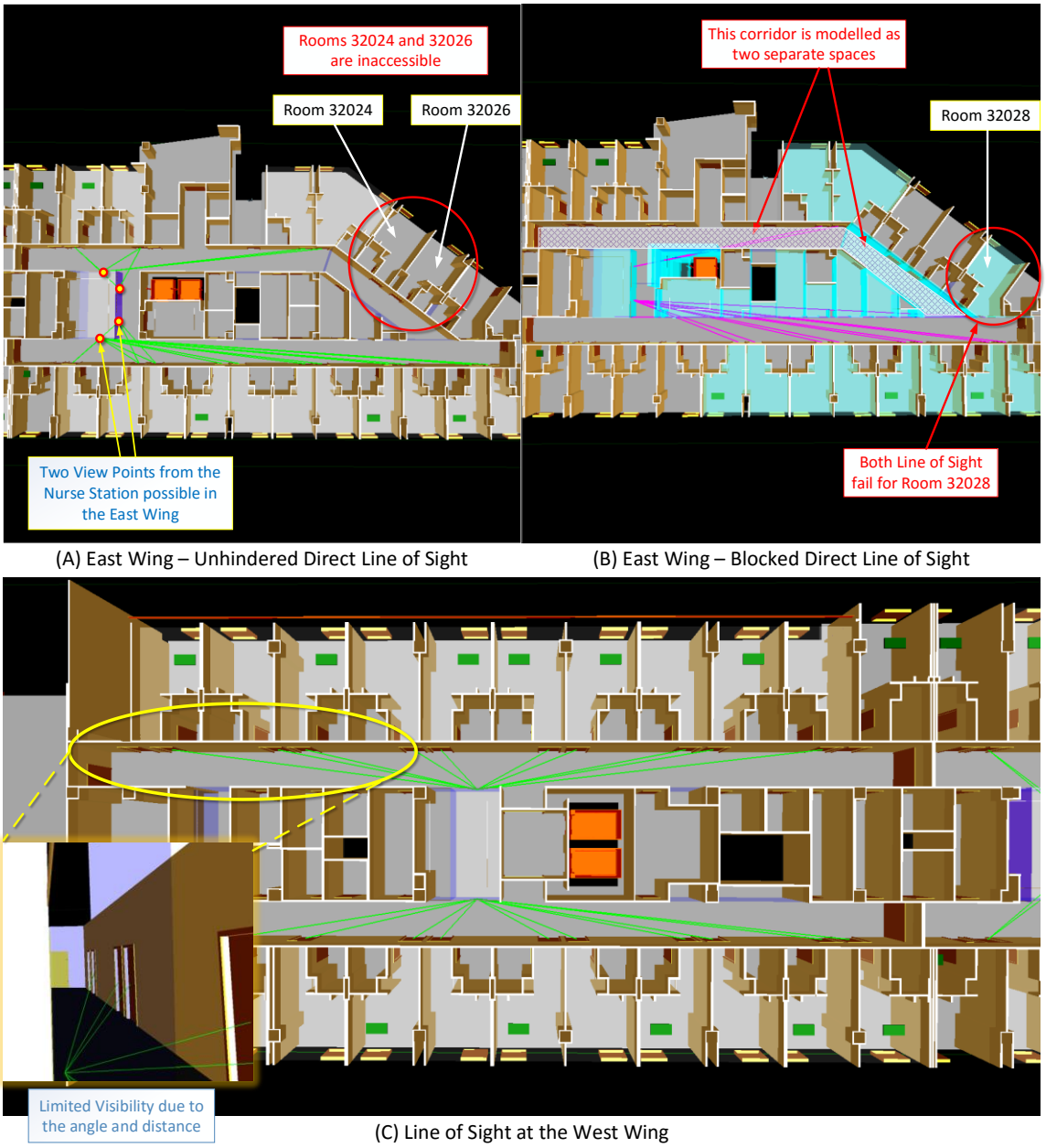
Table 17 - Two Rooms that Fail the Line of Sight Rule Check

Nurse Station Space ID	Patient Room Space ID	Patient Room No.	Patient Room Name
12gtomPxP5ChZwj0kVbOTe	3TX0KqOI1DKhxc0DcD2F5T	22028	M /S PATIENT ROOM
3MNNi1hTD0TQtfMnrUi_6j	3rTOFjmLr9FAv4PZreWVeE	32028	ISO PATIENT ROOM(-)

To ensure the complete coverage of the test, another rule can be run as described in section 8.2.3. This rule collects all Patient Rooms in the entire building and subtracts them with the set that is used in the main rule. It returns four rooms that are not accessible, and therefore “escape” the rule check, two at each level (Table 18). On closer inspection, they are connected to a corridor. However, this corridor is modelled as a space separated from the other part (Figure 105(B)). This separation is entirely artificial and it breaks the assumption that the nurse station must share a common corridor with the patient rooms.

Table 18 - Rooms that are Inaccessible from Any Nurse Stations

Space Element ID	Space No.	Space Name	Level
3TX0KqOI1DKhxc0DcD2F5L	22024	M /S PATIENT ROOM	2
3TX0KqOI1DKhxc0DcD2F5P	22026	M /S PATIENT ROOM	2
3rTOFjmLr9FAv4PZreWVeM	32024	M / S PATIENT ROOM	3
3rTOFjmLr9FAv4PZreWVeA	32026	ISO PATIENT ROOM(-)	3



(A) East Wing – Unhindered Direct Line of Sight

(B) East Wing – Blocked Direct Line of Sight

(C) Line of Sight at the West Wing

Figure 105 - Visual Report for Line of Sight Check on Level-3 of Model-E

Looking at the layout for the East Wing, even if the corridor is not split, these two rooms will still be without a direct line of sight from the nurse station. Therefore, they still fail the criteria set for this rule. This case raises an interesting issue regarding modeling standards. Currently such a standard does not exist. Without a standard, the rule checking result may be less certain. However, using the same BIMRL one may be able to define a dynamic query to the BIM data, to analyze BIM model and define new rule(s) as required to check such an issue. This is another very promising potential use of BIMRL. With such a feedback system, a loop process that starts with finding an issue, defining a new standard, and checking it to ensure adherence to the standard, is made possible with the assistance of the dynamic query feature in BIMRL. Only with the support of such processes can real progress towards process and quality improvement be ensured, measured and continuously updated to benefit any future projects.

9.1.2 Approximation of Actual Visibility Check

The second evaluation function calculates the estimated volume intersection between the view frustum from the nurse station and the patient room. The idea is to see in reality how much visibility the nurse will be able to see into the patient rooms from the nurse station. This is necessary because even with an unhindered line of sight, the actual visibility may be limited (Figure 105(C) inset). The Construct statement in the EVALUATE section generates the geometry based on a concept similar to view frustum. In this case the frustum is generated using the starting point at the nurse station location and connected to the opening area that faces the corridor. This is then extended following the ray lines from the starting point to form an extended frustum. The frustum is created as a Brep geometry that has volume. This volume is intersected with the patient room's volume. The computation is approximated by using their octree indexes that overlap. In this usage the approximate measurement is sufficient (Figure 106). The summary of the results is shown in Table 19. The result indicates that in most cases the actual visibility is

less than 5%. This appears to be too small for any meaningful view from the Nurse station. There are four pairs that have between 5 and 10%, four between 10 and 15% visibility, and only one with > 15% visibility (Figure 107). Upon closer inspection, even the sole pair with 15% visibility is not actually more than 15%. It is a false positive since the frustum volume into the room is made up by two lumps separated by the bathroom. The actual visibility probably still falls between 5 – 15%.

Table 19 - Summary Result from Visibility Rule Check

Viewable volume (%) range	Patient Room No
0%	22028, 32028, 31004
Less than 1%	31008, 21022, 32020, 22020, 31022, 21004, 21008, 32022, 21020, 22022, 32020, 31020, 31018, 21018, 31006
1 – 5%	21006, 32028, 32034, 32032, 32022, 22032, 22034, 32038, 32036, 22038, 22036, 21024, 31024, 32056, 32032, 32042, 22042, 32040, 22040, 31010, 32038, 22056, 32042, 21010, 32034, 31040, 22046, 21040, 32036, 32048, 31028, 31026, 32044, 21028, 31038, 21038, 32046, 21034, 31016, 21026, 32012, 21016, 32040, 22012, 22050, 32044, 31034, 32052, 22054, 32050, 32010, 32046, 22008, 22010, 32010, 31030, 31098, 21030, 21032, 31014, 21014, 22048
5 – 10%	32052, 22052, 31012, 21012
10 – 15%	21036, 31036, 32012, 32050
> 15%	32048

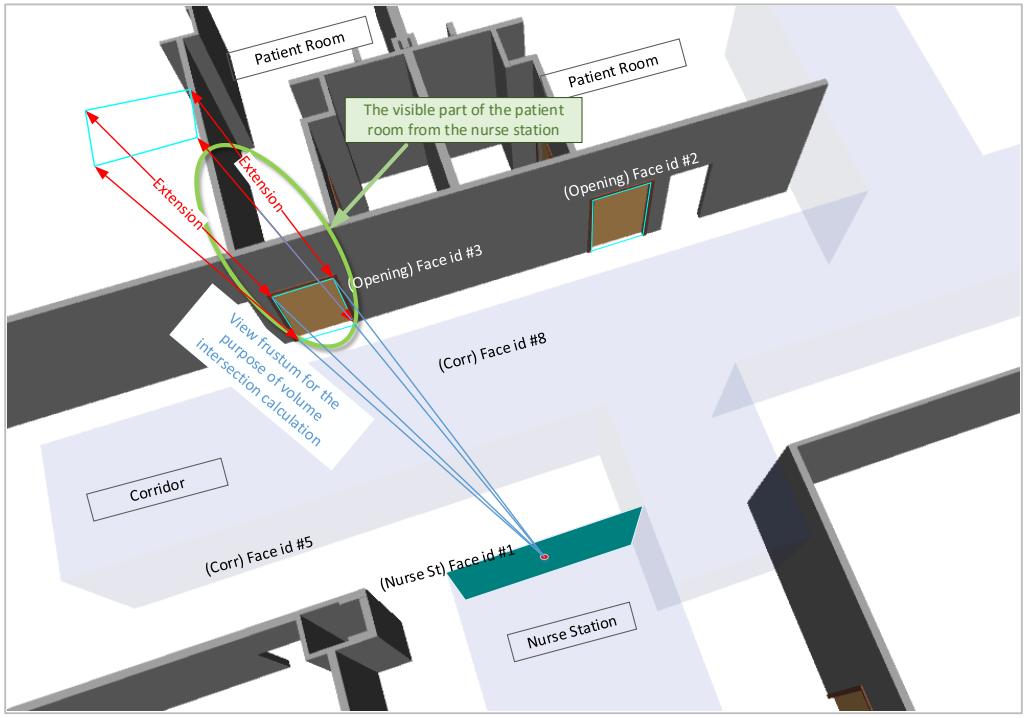
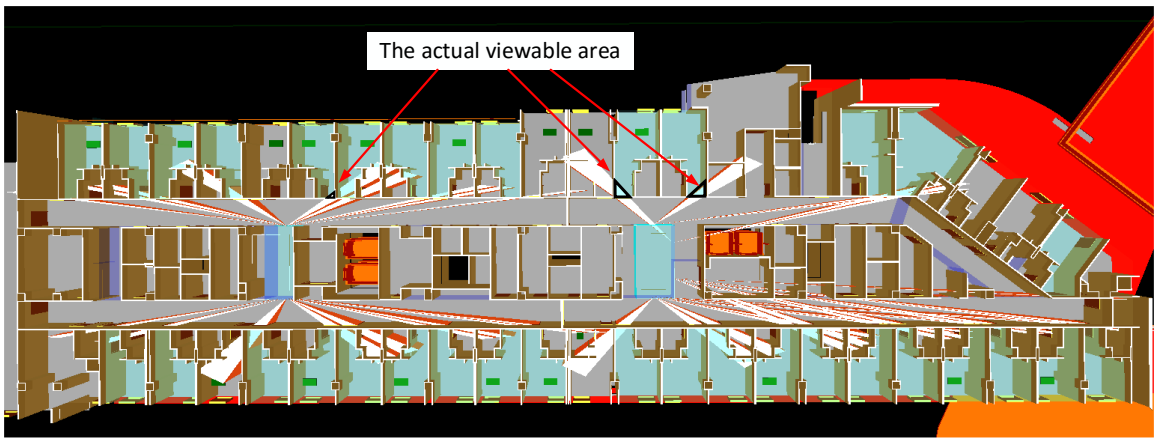
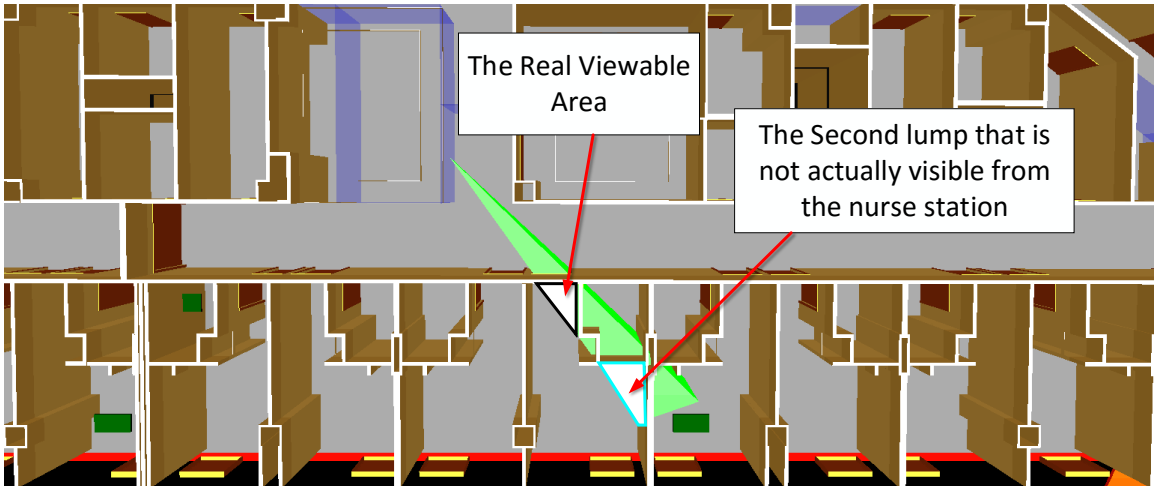


Figure 106 - Creating the Frustum for the Visibility Coverage Rule Check



(A) Viewable volume < 5%

(B) Viewable volume between 5% - 15%



(C) Viewable volume reportedly > 15% (false positive)

Figure 107 - Viewable Percentage Analysis from Nurse Stations to the Patient Rooms using View Frustum

9.2 Case #2 Test and Validation: Risk of Contamination

The rule is applicable to models that have MEP systems and at least one of these: a water tank, a transformer or a switchgear. One model that contains the relevant information (transformers) is selected to validate this rule. The model is the seven story conference building (Model-F). It is a medium sized building model that contains architectural elements and MEP elements in a single model. There are two transformers and one water tank.

Table 20 - The Main Objects of Interest for Case #2

	Water Tank	Transformer #1	Transformer #2
Element ID	1HYMcOjcD67RBbL8dgG1ue	3RTx5WcNX0cRLv7FpbvoXe	15E8uh5azAGPtKsV0k1U3M
IFC Element Type	IFCFLOWSTORAGEDEVICE	IFCENERGYCONVERSIONDEVICE	IFCENERGYCONVERSIONDEVICE
Name	A. O. Smith_Horizontal Mount Commerical Storage Tank_HD:A. O. Smith_Horizontal Mount Commerical Storage Tank_HD:1368069	3 Phase Dry Type Transformer:T-1:726386	3 Phase Dry Type Transformer:45kVA, 277 V/480 V, Three Phase, 4 Wires, Wye:1087102
Object Type	POTABLE WATER	45kVA	45kVA
Container ID	0uGek424j05BaSi7k8quVb	0uGek424j05BaSi7k8quVb	15E8uh5azAGPtKsV0k1U2l
Container Name	B120	B120	B146
Container Long Name	UTILITY	UTILITY	Electrical
Location	B1 BASEMENT	B1 BASEMENT	1ST FLOOR- ARCH
Type Name	A. O. Smith_Horizontal Mount Commerical Storage Tank_HD	45kVA	45kVA
IFC Type Object	IFCTANKTYPE	IFCTRANSFORMERTYPE	IFCTRANSFORMERTYPE

The model has been slightly edited to simulate the failed case for this rule by inserting a sanitary system with pipes passing through location above the Tank and Transformer #1 inside a concealed space above the ceiling (Figure 108).

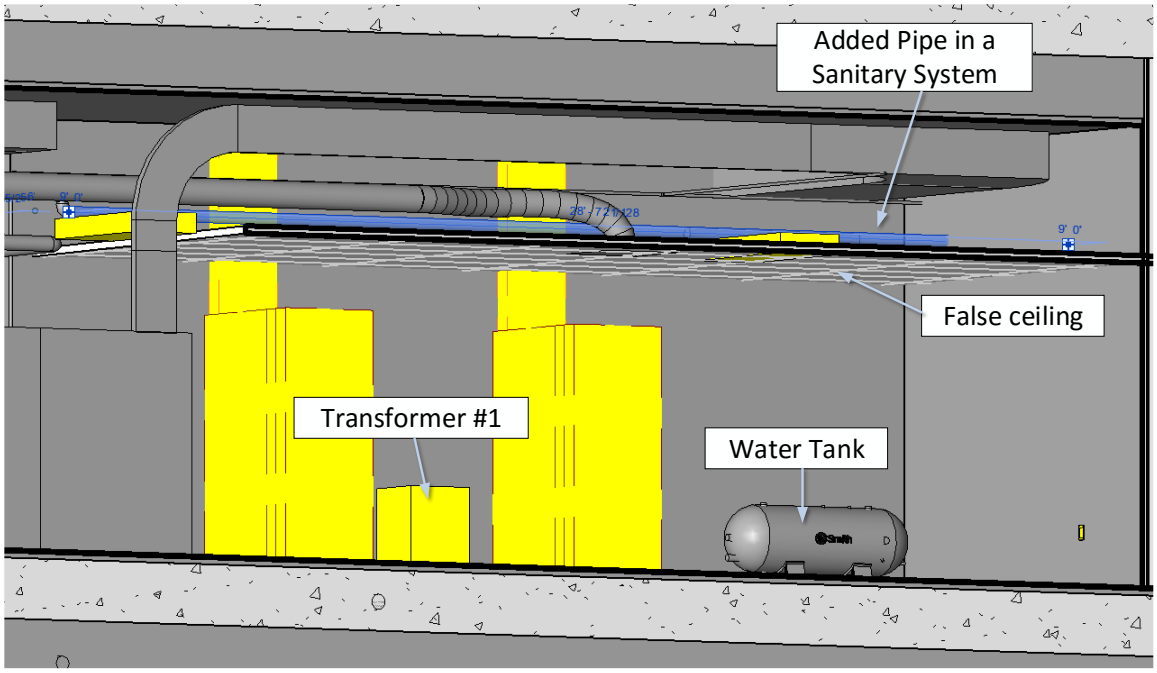


Figure 108 - Added Pipe in a Sanitary System

Applying the rule defined for this test case to the Model-F returns two objects that failed the test as expected, i.e. the water tank and transformer #1. Figure 109 shows the overview where the main objects being tested are and their collision check boxes. The two objects that fail the test together with the transient extrusion geometry created above them are shown in closer distance in Figure 110. The failure is reported because of a collision between these extrusion boxes and a pipe in the sanitary system overhead. Figure 111 shows the second transformer located at the 1st floor. This transformer does not detect a collision with any member of a sanitary system. BIMRL is able to discriminate between various members of MEP systems as they are already filtered in the second CHECK statement in Set2. The collision detection is only done between the transient extrusion geometries and the members of the sanitary systems. Note that this check depends on an additional property called ElevationHeight that keeps the height information of the top of a building story. It is currently non-standard in IFC since IfcBuildingStorey only has Elevation information as a direct attribute to IfcBuildingStorey. The ElevationHeight cannot be reliably derived from the data because

the story directly at the next highest elevation is not necessarily the expected height for that story.

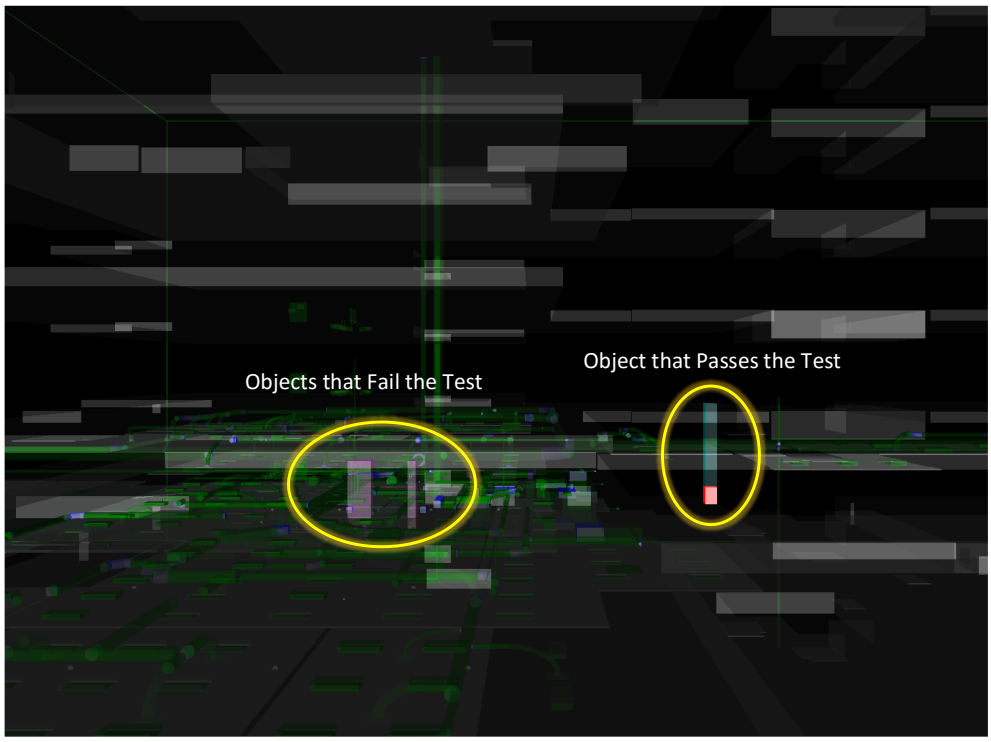


Figure 109 - Overview of the Visual Report of Test Case #2

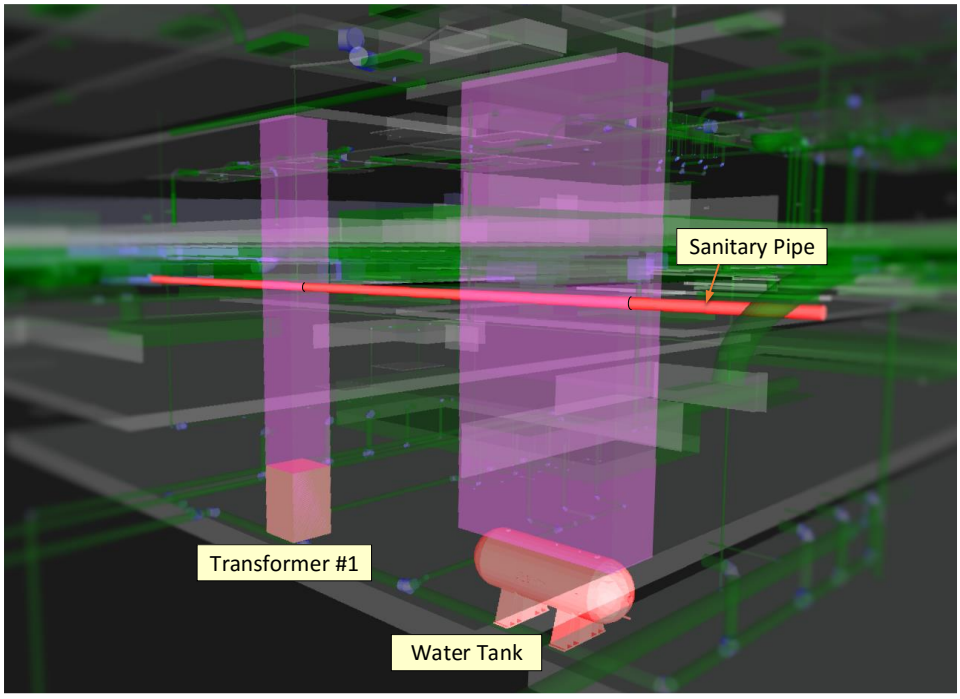


Figure 110 - The Objects that Fail the Test

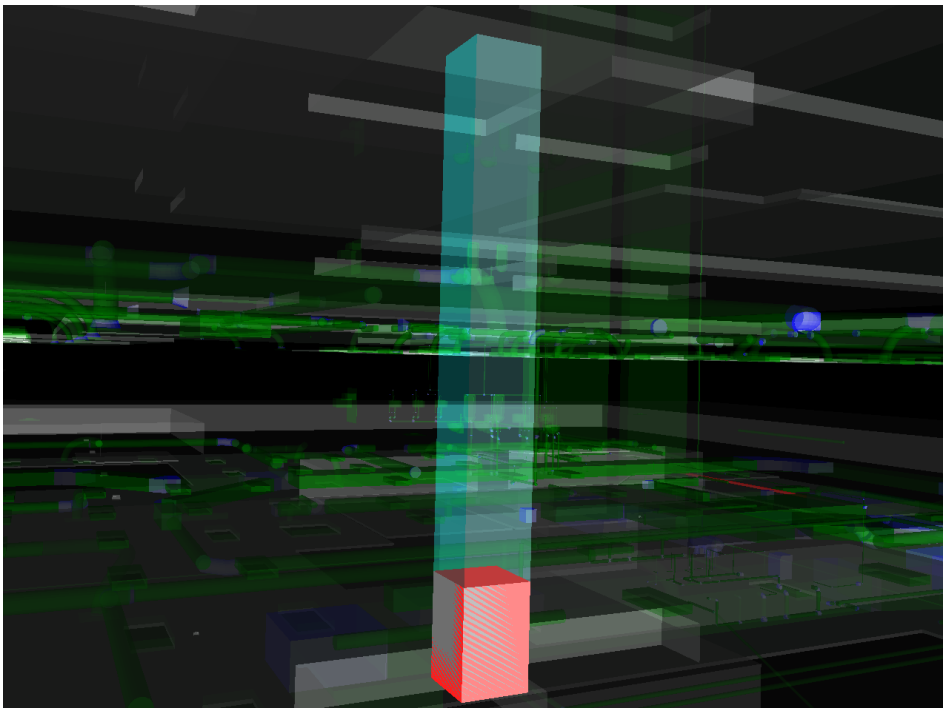


Figure 111 - The Object that Passes the Test

9.3 Case #3 Test and Validation: Remotely Located Exit Doors

Case #3 rule defined in the previous chapter is applied to two different models, i.e. Model-A, which is a single story school building, and Model-F. This rule is applicable to more models including hospital models, empty offices (at the design stage, before the interior is being designed) and any building used for assembly. To see both fail and pass cases, the models must contain at least one or more spaces that have an occupancy number or load of more than 50. These spaces require a minimum of two exits and they are to be located remotely from each other.

Ideally the model should contain relevant information in the form of:

- OccupancyNumber property in the property set Pset_SpaceOccupancyRequirements
- SprinklerProtection property in the Pset_BuildingStoreyCommon
- FireExit property assigned to the doors in Pset_DoorCommon

For the purpose of this exercise, the requirements above are relaxed by employing various conditions as described in the previous chapter when the rule is defined. In the two models that are suitable for this case, both involve a classroom or meeting rooms for relatively larger audiences. Model-A has two spaces with a seating capacity of more than 50, whereas Model-F has several spaces with capacities ranging from 60 to 100. The models have been modified a little to allow more possibilities for the check to find issues with the design, for example by reducing the number of exit doors to only one.

9.3.1 Results with Model-A

As the rule is implemented in two chained rules, the first rule collects all the spaces that qualify for the subsequent check for remoteness of two exit doors. For Model-A, the first rule returns two spaces that are "caught" based on the seating capacity directly counted from the number of chairs inside the spaces. This test uses geometry intersection that makes use of the spatial indexes. The two spaces are:

Table 21 - Two Spaces in Model-A that Qualify for Remotely Located Exit Doors Rule

	Space #1	Space #2
Element ID	1DBEX9khr8C9Qz6u$dcG4s	1DBEX9khr8C9Qz6u$dcG7E
Space Number	100	110
Space Name	MULTIPURPOSE SPACE_1	THEMED CLASSROOM
Seating Capacity (computed)	198	51

Figure 112 shows the multipurpose room (space no. 100) in the original BIM authoring environment. It has 198 chairs in total. Figure 113 shows one of the classroom (space no. 110) that has 51 chairs inside. The classroom has been slightly altered for the purpose of this test to simulate a fail case. It has been reduced to one door and the number of chairs have been increased to 51.

Table 22 and Table 23 list down the results generated after running the remotely located exits rule. There are many entries for the same room (space no. 100) because it

has many doors and the function reports every pair of the doors. To filter for an actual room that passes or fails, an additional SQL query can be used:

```
Select spaceid, spacename, spacelname, max(output)
   from remotelyloc2
   where object1 != object2
   group by spaceid, spacename, spacelname;
```

This query returns only one result for space no. 100, which passes the test because the maximum ratio between two of the exit doors is about 73%. This is one of 15 possibilities because this room has 2 external exit doors, 2 internal exit doors and 2 entrance doors that also serves as exit doors (Figure 115). Figure 114 shows the visual report generated from the rule. It is a combined view of pass and fail distances. Figure 114(A) shows Space no. 100 where the 15 possibilities of distances between the two exit doors are drawn. Green lines indicate that the distances fulfill the criteria. Red lines on the other hand indicates that the ratio is too small compared to the criteria. The red lines are harder to see because they are all along the wall or curtain wall since they are the lines connecting the exit doors along the same host wall or curtain wall. Figure 114(B) shows the other room (space no. 110). This classroom does not show any line for the distance between exit doors since it only has one exit door.

Elementid: 1DBEX9khr8C9Qz6u$dcG4s

Space no.: 100

Space Name: MULTIPURPOSE SPACE_1

Ratio: 0.728813317

Table 22 - Results with Ratio that Fail the Criteria for the Remotely Located Exits Rule

Space Element ID	Space No	Space Name	Ratio	Diagonal	Direct dist.	Door 1 Element ID	Door 2 Element ID
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0	88.87			
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.16	88.87	13.83	0nCSXJ5Zj82uihW87$iqVS	0nCSXJ5Zj82uihW87$iqVT
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.18	88.87	16.21	0nCSXJ5Zj82uihW87$iqVS	0nCSXJ5Zj82uihW87$iqVU

1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.34	88.87	30.02	0nCSXJ5Zj82uihW87$iqVS	0nCSXJ5Zj82uihW87$iqVV
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.34	88.87	30.02	0nCSXJ5Zj82uihW87$iqVT	0nCSXJ5Zj82uihW87$iqVU
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.18	88.87	16.21	0nCSXJ5Zj82uihW87$iqVT	0nCSXJ5Zj82uihW87$iqVV
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.48	88.87	42.50	1vjuPL4xX5Wf4plqg1YDN4	1vjuPL4xX5Wf4plqg1YDNA
1DBEX9khr8C9Qz6u$dcG7E	110	THEMED CLASSROOM	0	44.34			
1DBEX9khr8C9Qz6u$dcG7E	110	THEMED CLASSROOM	0	44.334	0	3IeFHlV7XAtwA12kBh6_Jf	3IeFHlV7XAtwA12kBh6_Jf

Table 23 - Results with Ratio that Pass the Criteria for the Remotely Located Exits Rule

Space Element ID	Space No	Space Name	Ratio	Diagonal	Direct dist	Door 1 Element ID	Door 2 Element ID
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.56	88.87	49.38	0nCSXJ5Zj82uihW87$iqVS	1vjuPL4xX5Wf4plqg1YDN4
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.62	88.87	55.00	0nCSXJ5Zj82uihW87$iqVS	1vjuPL4xX5Wf4plqg1YDNA
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.62	88.87	55.01	0nCSXJ5Zj82uihW87$iqVT	1vjuPL4xX5Wf4plqg1YDN4
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.56	88.87	49.38	0nCSXJ5Zj82uihW87$iqVT	1vjuPL4xX5Wf4plqg1YDNA
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.52	88.87	46.17	0nCSXJ5Zj82uihW87$iqVU	0nCSXJ5Zj82uihW87$iqVV
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.53	88.87	47.26	0nCSXJ5Zj82uihW87$iqVU	1vjuPL4xX5Wf4plqg1YDN4
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.73	88.87	64.77	0nCSXJ5Zj82uihW87$iqVU	1vjuPL4xX5Wf4plqg1YDNA
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.73	88.87	64.77	0nCSXJ5Zj82uihW87$iqVV	1vjuPL4xX5Wf4plqg1YDN4
1DBEX9khr8C9Qz6u$dcG4s	100	MULTIPURPOSE SPACE_1	0.53	88.87	47.26	0nCSXJ5Zj82uihW87$iqVV	1vjuPL4xX5Wf4plqg1YDNA

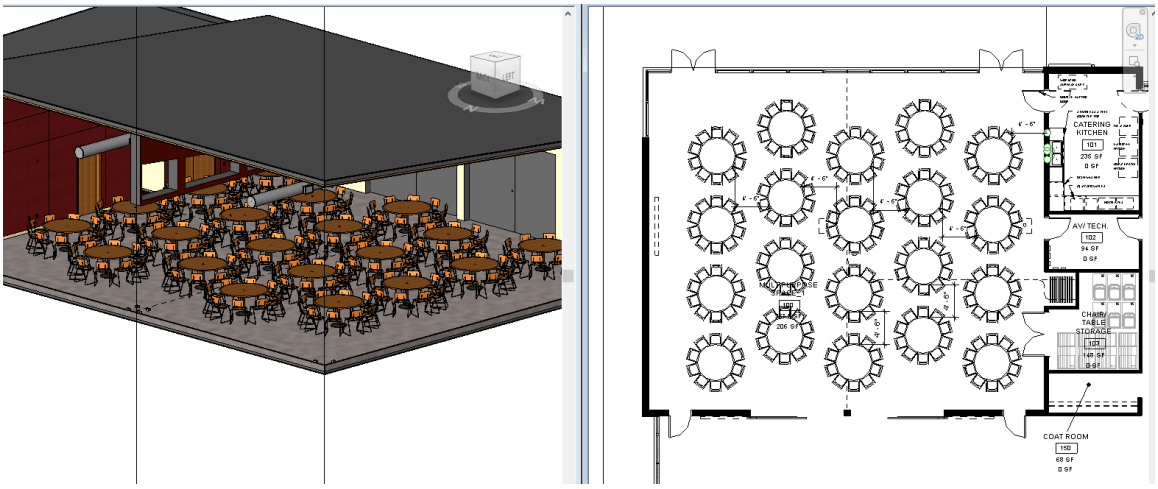


Figure 112 - Multi Purpose Space (space no: 100) in Model-A

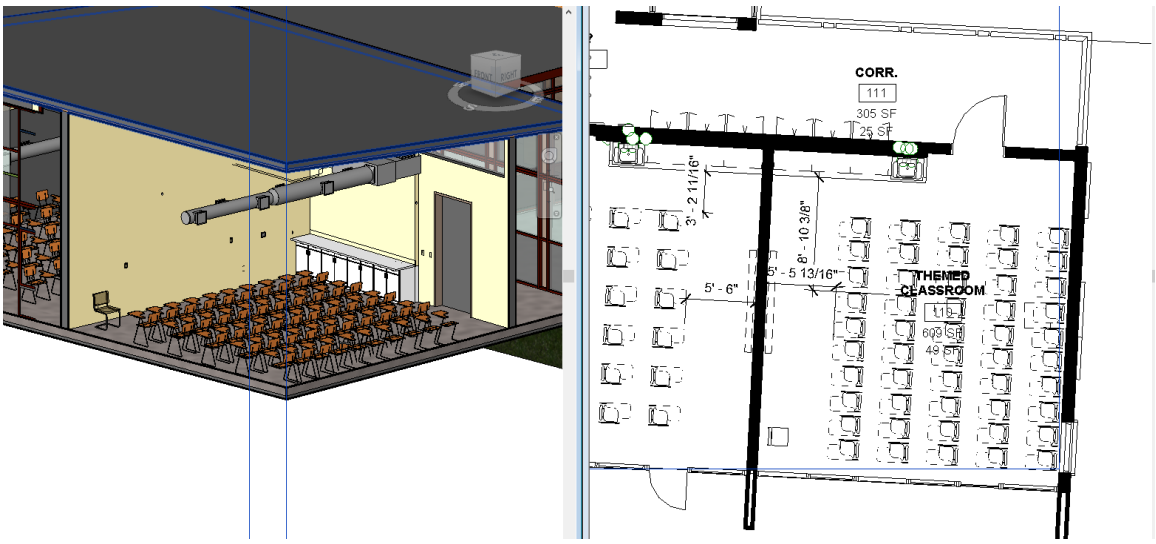
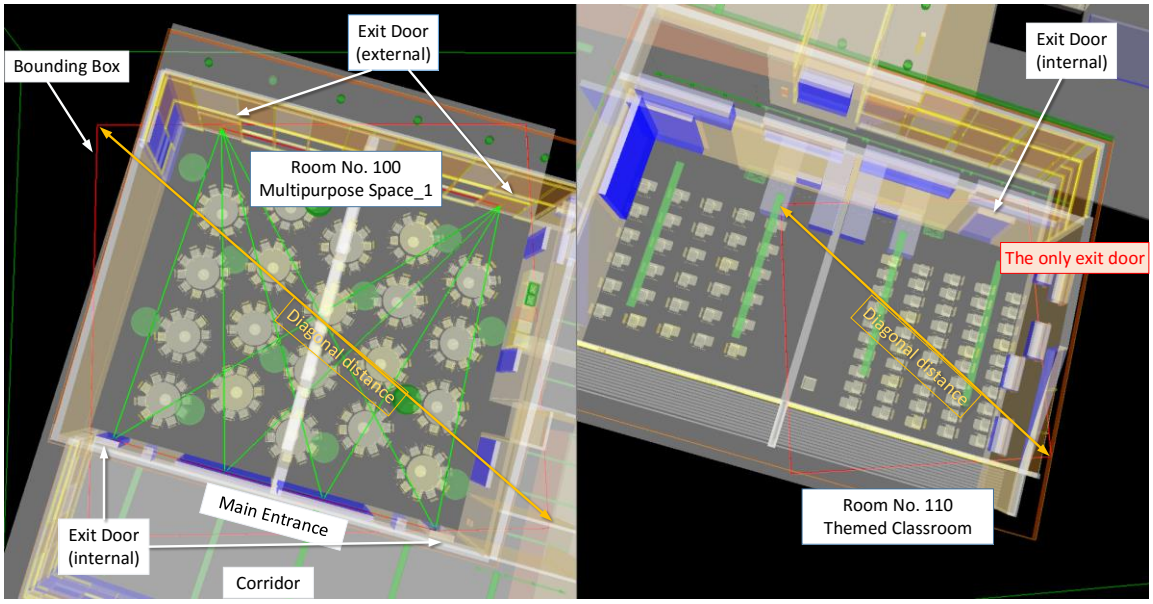


Figure 113 - Classroom (space no: 110) in Model-A



(A) Evaluation Result for Room No. 100 (B) Evaluation Result for Room No. 110

Figure 114 - Checking Result for Remotely Located Exit Doors on Model-A

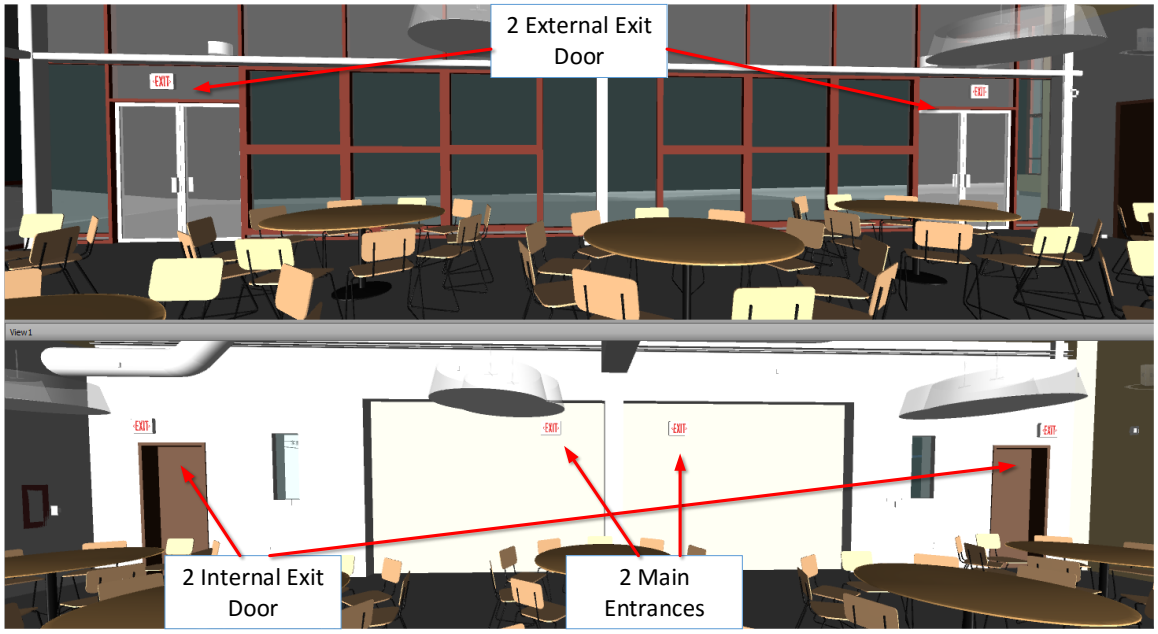


Figure 115 - The Internal View of the Exit Doors for Space No. 100

9.3.2 Results with Model-F

When the rule is applied to Model-F, the first rule returns a list of rooms with occupancy number from the property set, computed information such as the space area, seating capacity, and the number of sprinklers inside the room (Table 24). Table 24 includes redundant information because of the join from queries for the list properties mentioned earlier. The second rule is able to process the redundant information and using this result from the first rule, it returns the computed distance ratio for each pair of doors (Table 25).

Table 24 - Results from the First Rule of Test Case #3 for Model-F

Space Element ID	Space No.	Space Name	Occupancy No.	Story Name	Space Area	Seating Capacity	Sprinkler No.
1oeLNWdyvD2xBpFQTGTp9a	B118	CONFERENCE ROOM 1	100		1427.30		6
1oeLNWdyvD2xBpFQTGTp9a	B118	CONFERENCE ROOM 1	100	Default	1427.30		6
0SmY8owzL2ReIRyR3Lic4i	B133	CONFERENCE ROOM 2	60		857.49	60	
0PeKSAvI59PfDaeCq6ElAN	B132	CONFERENCE ROOM 1	60		857.49	60	7

0PeKSAvI59PfDaeCq6ElAJ	B134	CONFERENCE ROOM 3	60		857.49	60	7
0PeKSAvI59PfDaeCq6ElAH	B135	CONFERENCE ROOM 4	100		1219.47	96	
0SmY8owzL2ReIRyR3Lic4i	B133	CONFERENCE ROOM 2	60	Default	857.49	60	
0PeKSAvI59PfDaeCq6ElAN	B132	CONFERENCE ROOM 1	60	Default	857.49	60	7
0PeKSAvI59PfDaeCq6ElAJ	B134	CONFERENCE ROOM 3	60	Default	857.49	60	7
0PeKSAvI59PfDaeCq6ElAH	B135	CONFERENCE ROOM 4	100	Default	1219.46	96	
0uGek424j05BaSi7k8quU_	B109	CONFERENCE ROOM 3				60	
0uGek424j05BaSi7k8quUy	B108	CONFERENCE ROOM 2				60	

Table 25 - Results from the Second Rule of Test Case #3 for Model-F

Space Element ID	Space No.	Space Name	Distance Ratio	OBB Diagonal	Direct Distance	Exit Door 1	Exit Door 2
0uGek424j05BaSi7k8quUy	B108	CONFERENCE ROOM 2	0	43.40			
0uGek424j05BaSi7k8quUy	B108	CONFERENCE ROOM 2	0.42	43.40	18.12	3ZvKN1m6P2sOkrZo2giSPB	3ZvKN1m6P2sOkrZo2giSQh
0uGek424j05BaSi7k8quUy	B108	CONFERENCE ROOM 2	0.90	43.40	39.05	3ZvKN1m6P2sOkrZo2giSSB	3ZvKN1m6P2sOkrZo2giSPB
0uGek424j05BaSi7k8quUy	B108	CONFERENCE ROOM 2	0.76	43.40	33.11	3ZvKN1m6P2sOkrZo2giSSB	3ZvKN1m6P2sOkrZo2giSQh
0uGek424j05BaSi7k8quU_	B109	CONFERENCE ROOM 3	0	45.40			
0uGek424j05BaSi7k8quU_	B109	CONFERENCE ROOM 3	0.49	45.40	22.33	3ZvKN1m6P2sOkrZo2giSQT	3ZvKN1m6P2sOkrZo2giSRx
0uGek424j05BaSi7k8quU_	B109	CONFERENCE ROOM 3	0.91	45.40	41.46	3ZvKN1m6P2sOkrZo2giSQT	3ZvKN1m6P2sOkrZo2giSTD
0uGek424j05BaSi7k8quU_	B109	CONFERENCE ROOM 3	0.73	45.40	33.11	3ZvKN1m6P2sOkrZo2giSQT	3ZvKN1m6P2sOkrZo2giSTq
0uGek424j05BaSi7k8quU_	B109	CONFERENCE ROOM 3	0.72	45.40	33.11	3ZvKN1m6P2sOkrZo2giSRx	3ZvKN1m6P2sOkrZo2giSTD
0uGek424j05BaSi7k8quU_	B109	CONFERENCE ROOM 3	0.91	45.40	41.46	3ZvKN1m6P2sOkrZo2giSRx	3ZvKN1m6P2sOkrZo2giSTq

0uGek424j05BaSi7k8quU_	B109	CONFERENCE ROOM 3	0.61	45.40	27.85	3ZvKN1m6P2sOkrZo2giSTD	3ZvKN1m6P2sOkrZo2giSTq
1oeLNWdyvD2xBpFQTGTp9a	B118	CONFERENCE ROOM 1	0	61.39			
1oeLNWdyvD2xBpFQTGTp9a	B118	CONFERENCE ROOM 1	0.46	61.39	28.12	2XYrmJPhr0YeXiQuTxFowV	2XYrmJPhr0YeXiQuTxFowb
1oeLNWdyvD2xBpFQTGTp9a	B118	CONFERENCE ROOM 1	0.23	61.39	14.08	2XYrmJPhr0YeXiQuTxFowV	2XYrmJPhr0YeXiQuTxFowf
1oeLNWdyvD2xBpFQTGTp9a	B118	CONFERENCE ROOM 1	0.23	61.39	14.04	2XYrmJPhr0YeXiQuTxFowb	2XYrmJPhr0YeXiQuTxFowf
0PeKSAvI59PfDaeCq6ElAN	B132	CONFERENCE ROOM 1	0	43.27			
0PeKSAvI59PfDaeCq6ElAN	B132	CONFERENCE ROOM 1	0.72	43.27	31.40	0jdEFTa3r2xhGxvqi4$Tjn	0jdEFTa3r2xhGxvqi4$TiD
0SmY8owzL2ReIRyR3Lic4i	B133	CONFERENCE ROOM 2	0	43.27			
0SmY8owzL2ReIRyR3Lic4i	B133	CONFERENCE ROOM 2	0	43.27	0	0jdEFTa3r2xhGxvqi4$Tc9	0jdEFTa3r2xhGxvqi4$Tc9
0PeKSAvI59PfDaeCq6ElAJ	B134	CONFERENCE ROOM 3	0	43.27			
0PeKSAvI59PfDaeCq6ElAJ	B134	CONFERENCE ROOM 3	0.72	43.27	31.40	0jdEFTa3r2xhGxvqi4$TiW	0jdEFTa3r2xhGxvqi4Tc
0PeKSAvI59PfDaeCq6ElAH	B135	CONFERENCE ROOM 4	0	52.86			
0PeKSAvI59PfDaeCq6ElAH	B135	CONFERENCE ROOM 4	0.46	52.86	24.55	0jdEFTa3r2xhGxvqi4$TbW	0jdEFTa3r2xhGxvqi4$Tfr

Since the results in Table 25 still includes multiple results that are computed for each pair of exit doors, there are some cases that pass and some that fail. To consolidate the result, a simple SQL query can be used to show which space passes the test and which does not. The SQL query is as follows and the result of the query is given in Table 26.

```
SQL select * from
        (select spaceid,spacename,max(output)
              from remotelyloc group by spaceid,spacename)
  join (select spaceid,spacename,max(sprinklerno)
             from spaceoccupancy group by spaceid,spacename)
```

```
using (spaceid, spacename);
```

Table 26 - The Final Results for Remotely Located Exit Doors on Model-F

Space Element ID	Space Name	Distance Ratio between Two Exit	No of Sprinkler
0uGek424j05BaSi7k8quUy	B108	0.8998576679455150	
0uGek424j05BaSi7k8quU_	B109	0.91308444475493	
1oeLNWdyvD2xBpFQTGTp9a	B118	0.4581511706366360	6
0PeKSAvI59PfDaeCq6ElAN	B132	0.7255753406624320	7
0SmY8owzL2RelRyR3Lic4i	B133	0	
0PeKSAvI59PfDaeCq6ElAJ	B134	0.7255754834871890	7
0PeKSAvI59PfDaeCq6ElAH	B135	0.4643728371794130	

From the results reported in Table 26 Rooms no. B133 and B135 fail the test. Room B133 fails because there is only one exit door even though there are two other doors, but they do not lead directly to the circulation space. Room B135 has two exit doors that connect to the circulation space as required, but fail because the distance ratio between the exit doors is less than the required $\frac{1}{2}D_{diagonal}$ for a non-sprinkler protected space (Figure 116).

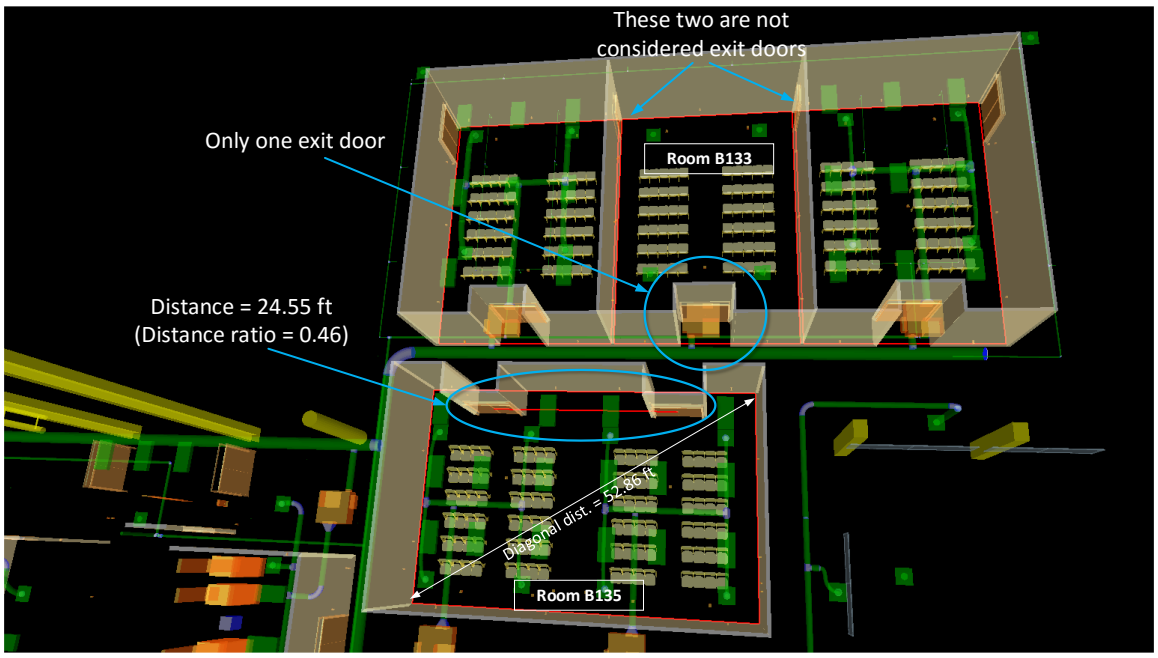


Figure 116 - Rooms that Fail Remotely Located Exits Rule Check in Model-F

The other rooms that have occupancy number above 50 have multiple exits. While some of the direct distances between any of two exit doors may fail, there are other pairs that pass the test. For example (refer to Figure 117):

- Room B118 has three exit doors the extreme left and right doors fulfill the requirement of $> \frac{1}{3}D_{diagonal}$ because of the sprinkler protection. Without that it will also fail the test.
- Room B108 has three exits and two out of three combinations fulfill the requirement of $> \frac{1}{2}D_{diagonal}$.
- Room B109 has four exits and five out of 6 possible combinations fulfill the requirement.

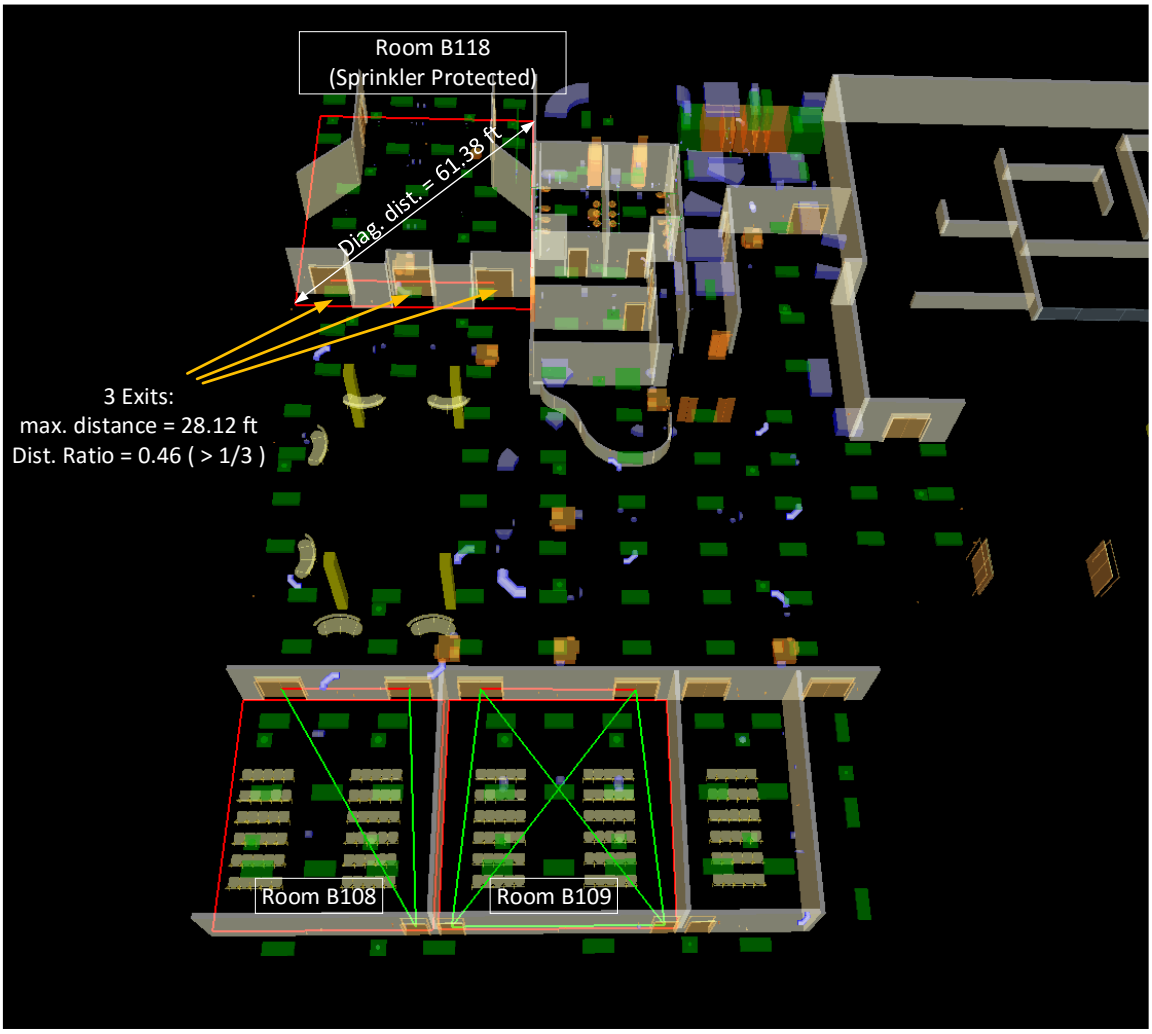


Figure 117 - Rooms that Pass the Remotely Located Exit Rules on Model-F

9.4 Case #4 Test and Validation: Accessibility Path

Executing the BIMRL statement as described in Chapter 8.5 on Model-E results in all 80 Patient Rooms being accessible from Vestibule 12048 (the starting point) through Elevators because the access using exit staircases has been explicitly specified to be avoided (vestibule 12003 is a vestibule for the exit staircase). Access to the patient rooms can also be through the staircase from that vestibule since the staircase is not explicitly excluded in this first example (Figure 118). There is the third vestibule (13002) that does not have any access to the patient rooms because it is located in a separate building from the main building where the patient rooms are (Figure 119).

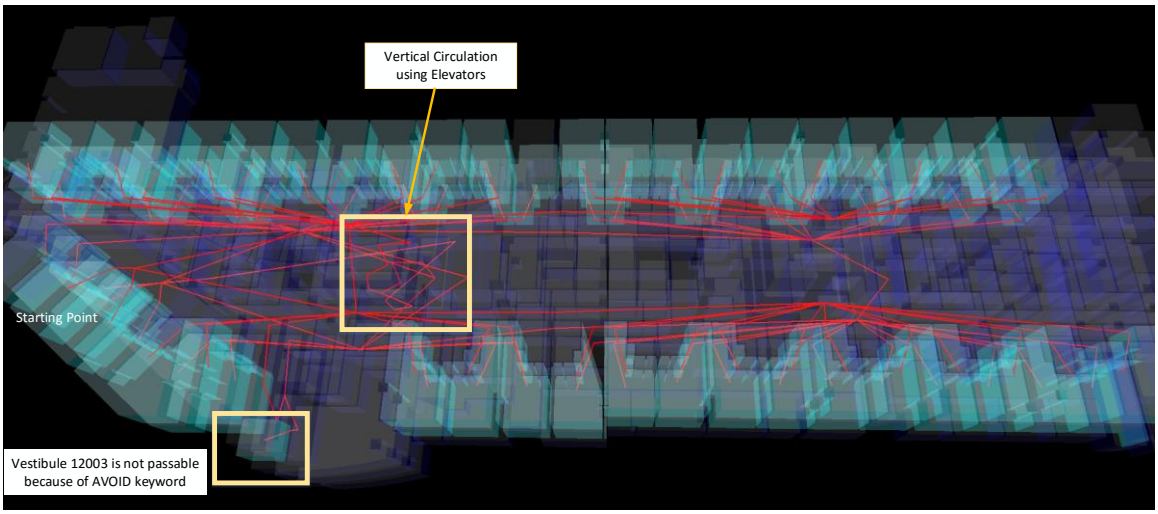


Figure 118 - Case #4 Accessibility to Patient Rooms through Elevators

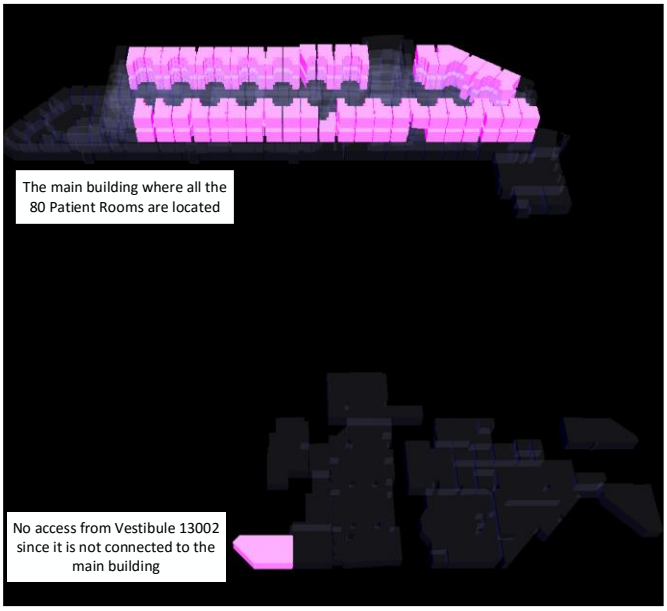


Figure 119 - Case #4 No Access to Patient Rooms from Vestibule 13002

Applying modified rules as described in chapter 8.5 such as additional variations numbers 1 and 2 will give different results:

1. Simulating egress path where no elevator should be used in case of emergency and no path through kitchen is permitted, the access to the patient rooms is only through staircases (Figure 120).

2. Specifying preference to use elevators for normal use or disable access and avoid the use of staircases, will result in access only through the elevators (Figure 121)

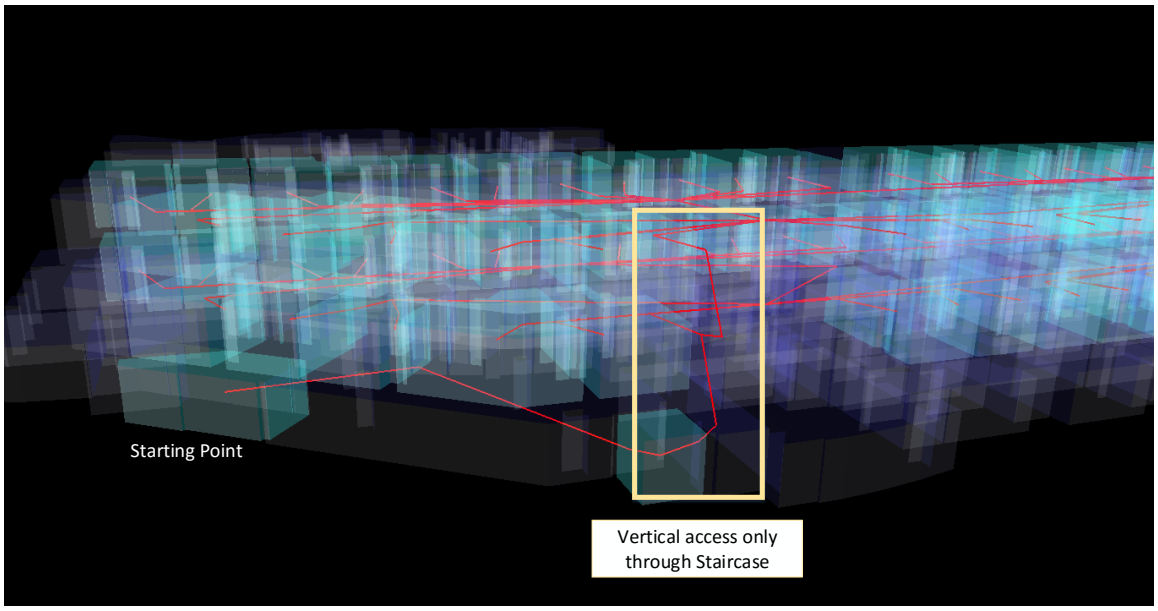


Figure 120 - Variation 1 of the Case #4 where elevator use is excluded and kitchen is to be avoided

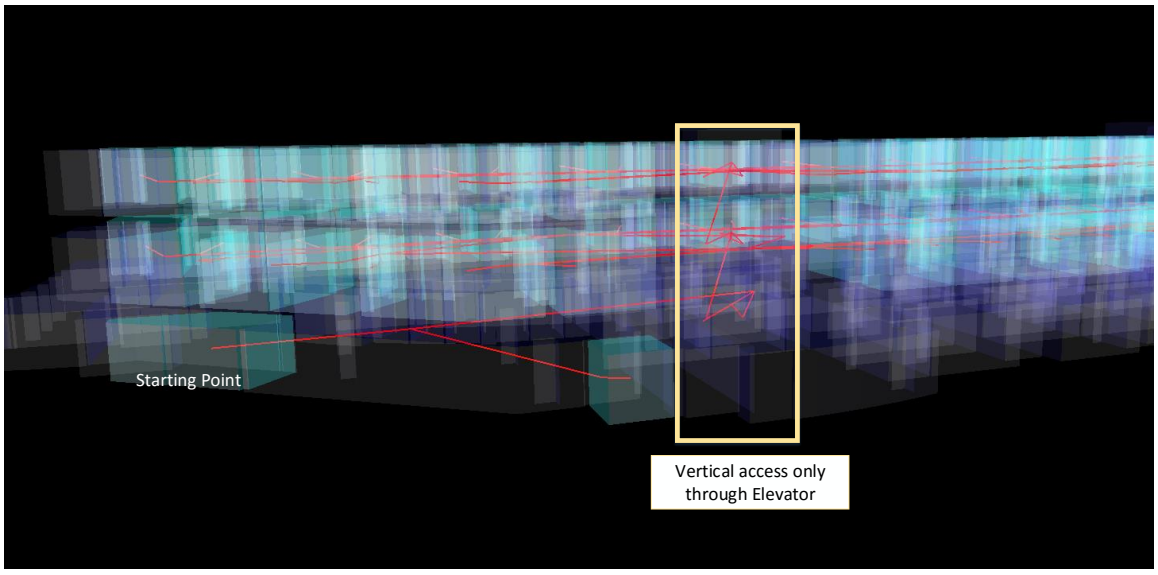


Figure 121 - Variation 2 of the Case #4 where elevators are preferred for normal use or Disable access

9.5 Discussions on Model Adequacy

Model adequacy has been observed as one of the most disruptive issues in a rule checking system. A well-defined model allows rules to be defined with higher precision and certainty. There are several ways to help reduce the uncertainties. Well-defined MVDs are a big help. Isolating the issues into an application layer may also achieve certain degree of certainty [45]. However, some of the issues of model adequacy need to be defined at the basic IFC definition, while others must be defined at the implementation and the user modeling levels. The latter can still be covered by MVD, but they are usually beyond the current tool for MVD checks due to the higher level of semantics involved. Several issues related to model adequacy are listed below. They are grouped into four categories. The list is far from being exhaustive, they represent only some samples that are encountered in the course of this research. The absence of well-defined model data will make rule checking extremely difficult and the results are uncertain.

9.5.1 IFC standard definition

1. One of the major inadequacies in an IFC model concerns a federated model. Current IFC models work best when everything is within a single model. This is not in line with the usual practice in real world projects where a project typically consists of multiple models federated together into one. Virtually all applications that perform such coordination rely only on the geometric co-location of the models. This may be sufficient for clash detection, but it is not sufficient for other applications that require more meaningful relationships to be adhered to, for example containment relationship, or space boundary. This issue has been addressed in a paper written in the course of this research [115]. The paper proposed a concept of deferred relationship that can be updated as the relevant data from another IFC file is added into the federated model. It can ensure that the

relationship data is maintained and connected across different files in a federated model.

2. Vertical relationship between building stories. In the current IFC definition, there is a lack of well-defined support for a relationship between floors. The only information available to use is the elevation attribute of the IfcBuildingStorey entity. The elevation information may lead to incorrect assumptions in several real-world situations that include:
    a. Multiple definitions of building stories from different models being federated. Some of them may carry slightly different information that make it difficult to reliably derive the proper information
    b. Existence of a mezzanine floor in a building story
    c. Varying levels of floor elevations in some buildings such as carpark decks

    A better definition of building stories and their relative position and a means of connection are needed. For example a definition that considers what is immediately above a building story and what is immediately below. This issue is encountered in the proof-of-concept case 2 above. This issue is also applicable to connections between buildings that require similar support.

9.5.2 Implementation standards

There are several issues that may have appropriate support in the IFC definition, but they are lacking in terms of implementation standards. They include:

1. Modeling space. Spaces in buildings often include concealed spaces above the ceiling or under the elevated floor. Should the concealed space be modelled as a separate space, as a partial space aggregated to the main space, or should it be modeled across the entire connected space independent of the actual space below it?

2. Multiple buildings. At present, there is no known BIM authoring tool that provides a good support to define multiple buildings. For a proper evaluation of connectivity or circulation, each of the graphs must be defined within its appropriate building stories and buildings at the higher level. Without proper building modeling, the circulation graph may become ambiguous.

9.5.3 Modeling standards

Other issues may fall into the category of modeling standards. There are currently various modeling standards written in different countries. It is still unknown how effective they are in practice and how widely adopted they are. Generally the current modeling standards may not be detailed enough to cover certain issues that may affect specific applications. In this research there are two issues encountered that can be overcome if the modelling standards are defined and used.

1. Splitting a common corridor. This issue has been described in case-1 above. It is legal to split a corridor into two or more, but such a decision may affect applications downstream. Such decisions should be standardized perhaps not necessarily on a national level, but could be within a project. This could complement the MVD definition, but it will require a more sophisticated checking tool to ensure compliance. The use of BIMRL may aid such checking requirements.
2. Modeling external spaces. Some of the models connect different levels of the buildings directly to the open space outside the building (e.g. Model-F). There needs to be a standard as to how to approach such models, as this will affect circulation graphs as well as fire exit requirements. If the external space is required, there has to be a well-defined identification to separate it from regular rooms/spaces. The use of a classification code may be the most appropriate

method to use that is better than relying on naming conventions, which is often less reliable.

3. Modeling elevator. Elevators, along with staircases, are one of the means to connect between building stories. From the various models used in this research, there is no uniform standard as to how this object should be defined. In some models, only the spaces representing the shaft is modelled. In others, an IfcBuildingElementProxy object is modeled, sometimes along with the space, and sometimes not. In one model (Model-E), the elevator is modeled at every level without spaces. In Model-C, it is modeled once at the ground level without a space, and in Model-B the void is modeled as a space without any elevator object.

9.5.4 Quality of the geometry data

Throughout this research, the quality of the geometry data constantly surfaces as one of the major hurdles for a good and consistent rule checking system. The problem seems to be a combination of software implementation and user modeling. One of the important objects required in the rule checking system is the Space object and therefore the well-formedness of the space objects with clear and correct boundaries and their relationship is critical. Any issue of the geometry of this type is difficult to detect since the geometry can be consistent in term of its structure, but yet be incorrect. Such issues threaten the usefulness of the model for various applications downstream. Attempts to reconstruct some of the relationships in this research often met with unreliable outcomes because of this issue. Paper [50] has proposed a method to measure the geometry quality by comparing the original data and the exported (and imported) geometry, but more may need to be done to address other issues. Some of the issues experienced during this research are given below. Note that the examples are from Autodesk Revit, but they may also be applicable to other BIM authoring tools.

1. Space geometry bleeds into unintended areas

   When boundaries of the space object are not closed properly, the space geometry may bleed into unintended areas and cause the geometry to be incorrect even though the data structure may be consistent. Figure 122 shows an example of such a case. This case often resulted because the supposed boundary is not air-tight. While this may be largely a modeling issue, the software implementation should highlight such problematic areas. It will be impossible to detect any issue in the downstream applications such as rule checking systems.

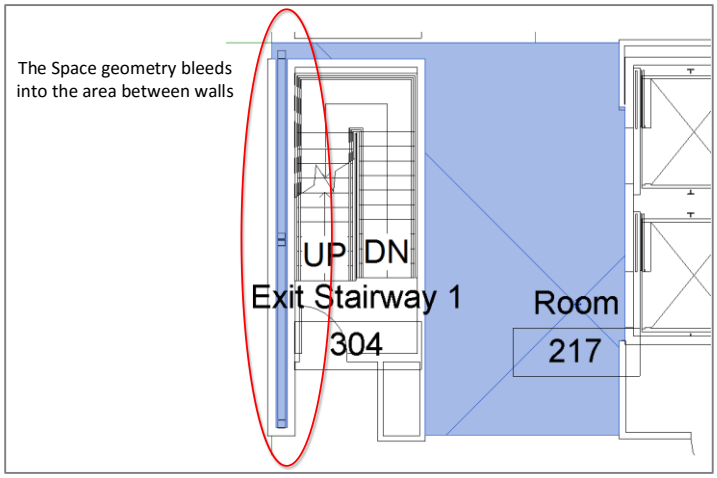


Figure 122 - Space Geometry Bleeds into Unintended Area

2. Boundary object without a proper boundary

   Certain objects such as Curtainwall in Revit appears to define a boundary for space bounding algorithms as its centerline (Figure 123). This may cause a problem in the detection of a virtual space boundary because the Curtainwall centerline will be shared by the two spaces on both sides of the curtainwall, thus creating what appears to be a virtual space boundary.

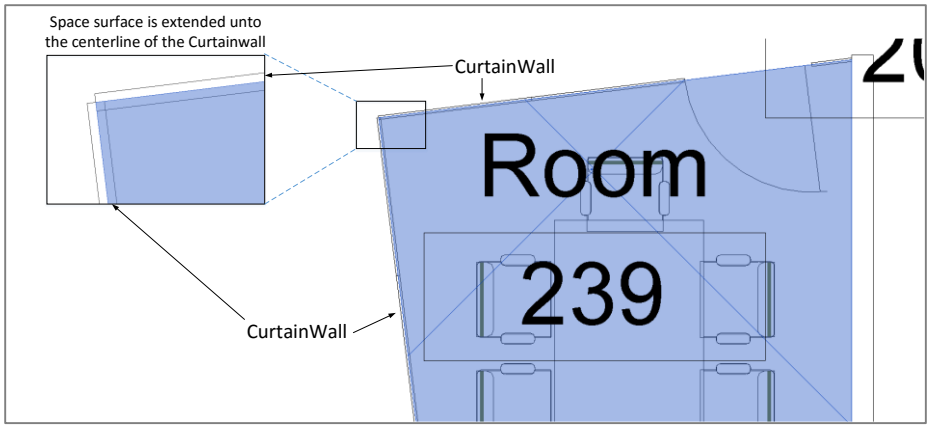


Figure 123 - Boundary Object without a Proper Boundary

3. Inaccurate virtual space separation

Using a space separation line in Revit seems to be a haphazard exercise since it may cause all kind of issues at the joints. Figure 124 shows an example of this. This may be a combination of user modeling and software implementation issues. This model may cause the appropriate link between the two circulation spaces in this example to be missing.

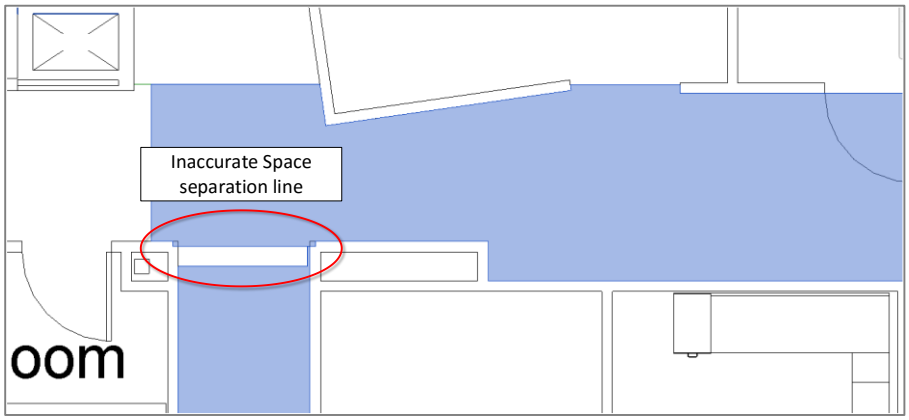


Figure 124 - Inaccurate Virtual Space Separation

4. Geometry precision

Similar to the situation above, there are situations where the space geometry will be affected by how the corners influence the space boundary representation. In the example shown in Figure 125, one space surface is split into

two different faces with different normals because of the differences between corners at the bottom and at the top resulting in non-coplanar faces. These faces cause issues when they are supposed to be a virtual boundary between one space and the other.

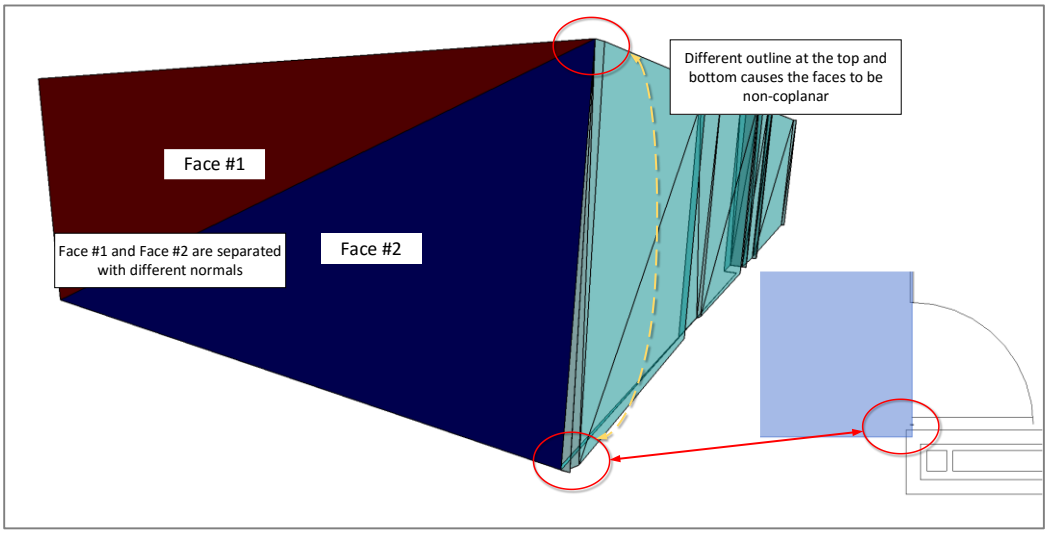


Figure 125 - Geometry Imprecision Causes a Space Surface to be Split

The above examples highlight the urgency to improve quality check for basic information in the model including the geometry. This is needed both at the very basic level of geometry correctness and at the higher level of semantics such as virtual space boundary and air-tightness check. Such tests should be considered to be part of future IFC certifications.

9.6 Assessment of BIMRL as the answer to the research questions

The three major concepts of BIMRL, i.e. the simplified schema, multiple geometry representations, and the rule language, have been validated in the proof-of-concept described in CHAPTER 8 and CHAPTER 9. The validation has provided

answers to the research questions that are described in section 1.2.2. The answers can be summarized as follows:

1. There is no single answer to the question regarding the suitable approach to the complexity of rule checking and its large number of variations. From the analysis of the complexity of rules from the implementation perspective (CHAPTER 3) and the analysis of the rule structure (CHAPTER 4), it is clear that the best strategy to deal with the complexity is first to recognize the need for the essential components that will support the implementation as described in each of the class of rules. By providing the essential components as a standard facility in the rule checking system and making it accessible to the rule checking language by means of integrated queries and simplified access using the multiple representation concept, the complexity of the rule implementation can be significantly reduced as demonstrated earlier in this chapter. All of which do not require a heavyweight geometry engine or modeler, nor a dedicated graph engine. Yet, it is still able to solve relatively complex rule requirements. The second answer is to provide a language that can be adapted to various requirements. BIMRL as a language provides a framework that allows for a wide range of rule requirements to be defined using just one or a few rule definitions. Since the language also supports variables, it provides the capability to support parameterization of rule definitions. The language also allows different levels of granularity by how much checking logic is embedded into the evaluation function as opposed to explicitly stating it in the definition itself. Nevertheless, it comes with a tradeoff between simplicity and granularity or generality of the rule definition. The author believes that these are the best ways in addressing complexity of the rule requirements and their number of variations.
2. The answer to the second question is more precise with the simplified star-like database schema combined with the ability to store both geometry and graphs in

the same database. The simplified schema provides the answer to two big questions, i.e. simplicity in accessing the data, and the efficiency in querying and obtaining the data.

3. There are many approaches that look attractive in translating rules into their computable forms. However, they generally do not account for the issue of dealing with where the data should be obtained and how to obtain them. Rules generally contain non-explicit semantic concepts that will never be represented in the model. Pre-computing such information may not be practical unless the system is addressing a very narrow scope of requirements. The use of CG discussed in CHAPTER 3 addresses this issue by treating rule requirements as knowledge. The CG coupled with a simple process to interpret the rule, will expose the higher level semantic concepts that are usually translated into derived entities and functions. By defining the derived entities and functions, rule requirements minimize ambiguity and specify where and how the data should be obtained.
4. BIMRL grammar is designed to allow query-based rule definitions with a wide range of capabilities to deal with simple to complex rule requirements. With the acknowledgment that it is impossible to perform every detail of the checking logic in the grammar, BIMRL provides an extension through a plug-in mechanism for the relevant evaluation function. This way, even a very complex requirement can be addressed with BIMRL without the need to modify the grammar. The dynamic parameter definition of the evaluation function interface allows for flexibility - what type of data and how much data should be expected by the function.
5. In CHAPTER 3, analysis of the rule structure shows that there are several essential features that a rule language should support, i.e. a query system, logic and relational operations such as join, aggregate, boolean, arithmetic, spatial and

graph. BIMRL supports them all and with an efficient query system based on a relational database system.

Generally, the validation of the proof-of-concept cases discussed earlier in this chapter demonstrates that concepts introduced by BIMRL provide answers to the research questions.

CHAPTER 10

CONCLUSIONS

Rule checking is a complex problem that involves various subject matters. They are outlined as follows [6]:

1. Converting rules written for manual interpretation into a form that is amenable for computer implementation; this is a major issue for legacy systems such as building codes; Rule languages may be defined that are relatively semantically unambiguous to both knowledgeable computers and people.
2. Defining the model view needed to support checking, in terms of the object, attributes and relations required to either directly check the data, or to allow the checking procedures to derive the data needed for checking.
3. Executing the rule, dealing with the logic of the rule, in terms of nested conditions, combinatorial conditions and the logical issues of existence tests versus universality tests.
4. Reporting back to interested parties the results of the rule checking, in a manner facilitating correction of identified issues.
5. Setting up procedures for automatically correcting rule failures.

This thesis has discussed and demonstrated the possibilities of solving a large portion of the above subject matters. The approaches that have been explored in this research offer an effective rule checking tool that could provide an integrated solution, which facilitates an automated rule checking system. In chapter 3, an elaborate analysis to identify high level computational capabilities needed to support rules of various complexity has been presented. Chapter 4 demonstrates a method, procedure and tool to capture rule interpretation into a systematic, unambiguous, computer and people friendly form of knowledge representation using the Conceptual Graph. This is a direct solution to

deal with items 1 and 2 in the list above. Items 3 and 4 are addressed in chapters 5, 6 and 7, where an efficient query-able BIM database schema is defined together with its support for the geometry and spatial representations inside a database, and a BIM Rule Language that glues them into a neat language interface to implement the rules. The prototype system developed as part of the proof-of-concept has shown the effectiveness of such an approach (chapter 8 and 9). The prototype system shows significant potential to handle even relatively complex checking tasks that until today require significant programming effort and specialized knowledge. By defining a rule in a BIMRL triplet statement, one is able to define a rule that may range from a simple rule (class-1) to relatively complex ones (class-2 and 3). The language structure through the CHECK-EVALUATE-ACTION triplet allows for flexible filtering, evaluation and reporting to be in done in one step. This allows many rules to be automated with relative simplicity using just a single statement in most cases. A more complex form of rules can be done through various options:

1. By chained evaluation functions within a single BIMRL triplet statement. This is suitable when the objects being checked are closely related and may be shared in the chained evaluation functions. The selection(s) is done once and the evaluation functions are executed in sequence, usually one function produces results that the subsequent function uses as an input.
2. By chained rules. For more loosely connected rules, using multiple BIMRL triplet statements may be more practical and easier to manage, especially if the results can be used by multiple other rules. The BIMRL triplet allows the results of a rule to be permanently stored in database tables, so that it can be used as an input for any other subsequent rules. BIMRL rules do not strictly limit rule definition to an evaluation of a final pass or fail result only. It can be defined as an intermediate rule that can be used to define an intermediate state to determine whether a further check is applicable or necessary to be done for

a set of rules. The test case #3 demonstrates this capability, where the first rule is an intermediate rule that defines the applicable objects based on a series of conditions defined in the first rule.

3. By using an evaluation function extension facility. In extremely complex rules where sub-rules may be dependent on each other or for a very specialized checking logic that may be applicable to one type of checking, a generalized system usually is not sufficient. BIMRL allows extensions to be written and plugged into evaluation functions. Such extensions can be added without a need for changing the language structure and can be integrated directly into the rest of the facilities BIMRL supports.

BIMRL offers significant contributions to solve the longstanding issues with the automated rule checking problem. Its innovative approach in dealing with geometry data, query system and the language interface has been demonstrated. BIMRL has been validated as able to deal with complex rules that traditionally can only be done with a rather complex programmatic approach. The programmatic approach does not allow for flexibility of user-defined queries and it is not easy to extend the functionality to respond to the variations often required for rules and also not easy to define a new rule. BIMRL on the other hand offers a flexible query based checking language that allows user-defined criteria to be specified easily and an extensible evaluation function to be integrated when a more complex checking logic is required.

It is foreseen that BIMRL will open up various possibilities even beyond the traditional understanding of rule checking. It provides a tool to analyze and to gain insights into BIM models in a way that is hardly possible today without using a specialized tool. The list below shows a sample of such possibilities:

1. A simple extension to the MVD checking. The current MVD checking only works with explicit IFC schema and data. There are other MVD related

checks that involve semantically richer concepts within the building model that cannot be checked with the traditional MVD check. Some of these rules are discussed in [50], for example the consistency of containment relationship, completeness of space boundary information, etc.

2. Defining a more well-defined modeling standard and providing rules to ensure such standards are being adhered to. This may be applicable to the idea of a more precise definition of the level of development (LOD) that should be defined according to the needs of each individual project. Such definitions can only be effective if models can be validated against. BIMRL facilitates the definition of those rules and the check for their adherence.
3. Defining rules to check space programming requirements may be possible using BIMRL rather easily. It is not limited to just comparing room data sheets that can be captured into a database table, but also allowing computation of the approximate use of footprints by various furniture and appliances in the room. In addition, the spatial indexes in BIMRL overcome the limitations of federated models that are made of separate building models for various disciplines and derives a consistent containment relationship.
4. Integration with external simulation programs that currently require own, separate models (often simplified) such as CONTAM for air leakage analysis.
5. Building up a knowledge based system where issues that are identified and experience that are obtained in previous works can be captured, defined into rules and set up as new standards to ensure the same issue will not be repeated in future. This is a partial answer to item no. 5 in the list discussed at the beginning of this chapter.
6. Performing ad-hoc queries or analysis of the model that are not limited to just the properties but also geometry and spatial relationships. With the ability to

define transient geometry, it may be possible to explore a new way to analyze building models. For example exploring certain aspects of space syntax in 3D.

While the potential uses of BIMRL can be wide ranging, it will never be a one-size-fits-all tool for every imaginable issue. There are always limitations that such systems are never designed to handle. They include the following:

1. As mentioned in the beginning of this thesis, this system operates on one important basic premise that the data is read-only, or mostly for read-only purpose. It is never intended to handle an update. Any updates required should be done directly on the source of BIM data, i.e. the original BIM authoring tool. There may be some limited method of updates that can be automated indirectly through other means using a specialized exchange format. Such requirements should be treated very carefully and avoided if possible.
2. As any similar system, rule checking will only be meaningful given a complete building model. It should not be overburdened by the need to find out the completeness of the data. This should be done in a separate process prior to rule checking as part of an MVD test to ensure that the minimum data quality required has been met by the building model. BIMRL can also be used to complement MVD checking for requirements that involves richer semantics as mentioned earlier.
3. Complex rules that are often found in building codes may be broken down into their atomic rules and either run separately or defined in chained rules. Even with this, some rules simply require an extensive algorithm implementation that may be suitable just with a “fat” evaluation function extension.

While the prototype system developed in the course of this research is a complete and comprehensive prototype, is still just a prototype. There is more work to be done to

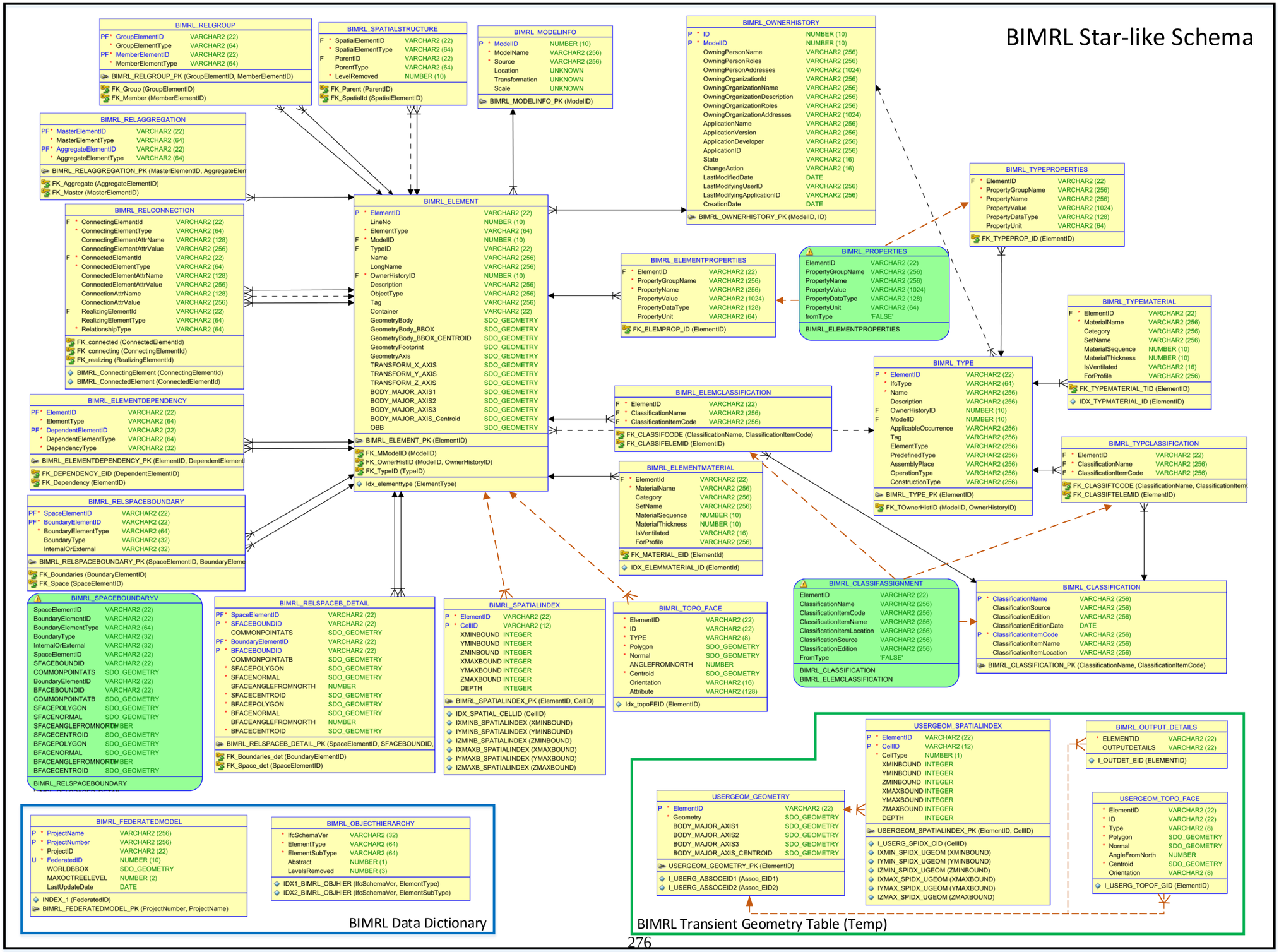
elevate it into a production ready application. The main work that remains to be done is in the area of ETL performance. The bulk of the time taken in the ETL process goes into computing the topology surfaces and the generation of the Octree spatial indexes for the entire building data at once. Deriving the topology surfaces from the source is probably the better way to do it. It will be less costly to do so and more reliable since there is a direct access to the source. Certain parallel computing techniques have been applied in generating the spatial indexes, but more optimization and parallelization can be done to reduce the time required to perform the ETL process. The prototype system is currently using an Oracle RDBMS. However there is only a shallow dependency on the Oracle database. It should be straightforward to migrate the database into other databases as long as there is a good mechanism to store the geometry polyhedra. This database includes the popular open source database PostgreSQL with PostGIS extension for storing the geometry data. With the increasing popularity of the cloud, there is also an opportunity to look into different underlying database uses beyond RDBMS to include specialized databases that are more suitable for the use in the cloud environment.

The author thinks that despite the limitations, BIMRL offers a significant breakthrough for automated code checking in that it reduces the gaps between various classes of rule complexity, and lowers down the barrier to perform rule checking without extensive programming. The availability of a query-able BIM alone already offers many interesting opportunities to uncover the data that may not be available today. The availability of the integrated geometry and spatial information, and the flexible rule checking language with BIMRL, will make many things that are difficult to perform today possible.

APPENDIX A

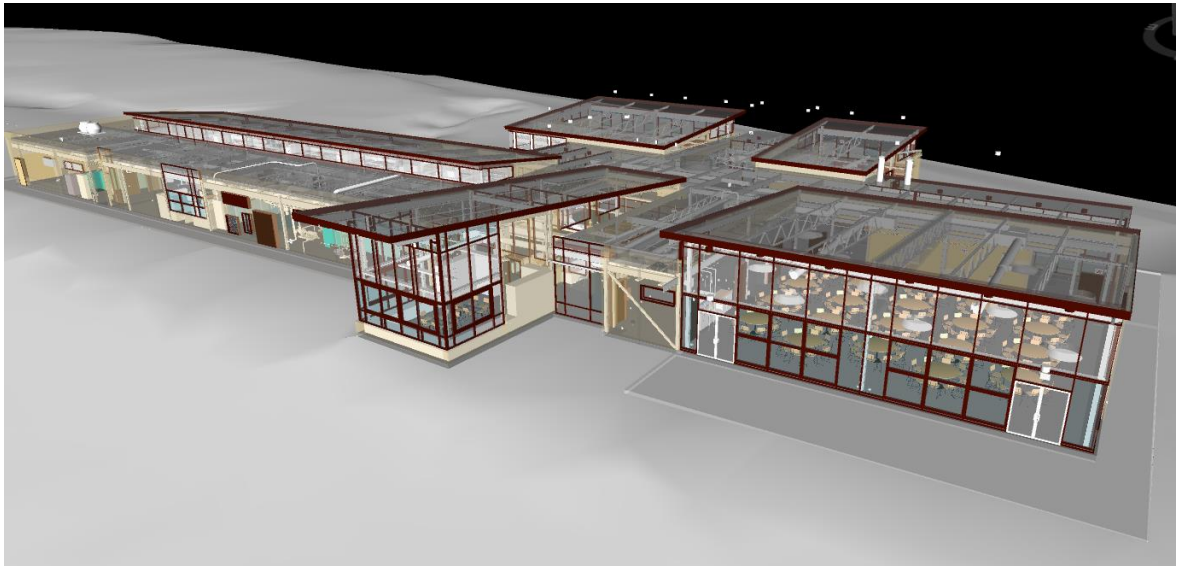
BIMRL SCHEMA DIAGRAM

BIMRL Star-like Schema



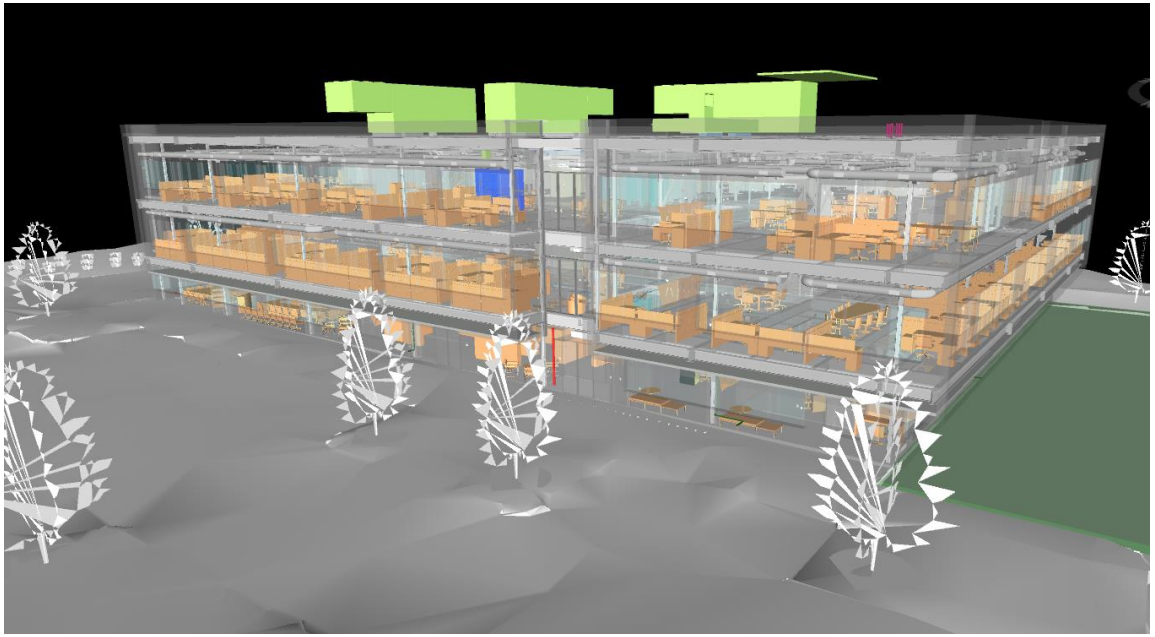
APPENDIX B

STATISTICS ON THE BUILDING MODELS USED

Model	Models inside the federated model	Counts of IFC entities	Counts of objects in BIMRL schema	
	Architecture	349,359 (31.6 MB)	Table	Count
Model-A:	MEP	503,425 (33.0 MB)	Elements	8,780
One storey school model	Structure	429,551 (22.4 MB)	Types	1,495
	***Total***	***1,282,335 (87 MB)***	Properties	235,602
			Relationships	15,104
			Others	5,067
			Spatial Index (Level = 10)	15,933,552
			Topology Face	1,639,288
			Total	***17,838,888***

Model-B:

Three storey office model



Architecture	383,048 (21.8 MB)
MEP	1,142,739 (66.0 MB)
Structural	53,733 (3.5 MB)
Total	***1,597,520 (91.3 MB)***

Table	Count
Elements	14,102
Types	2,068
Properties	168,316
Relationships	29,526
Others	4,875
Spatial Index (Level = 9)	11,802,657
Topology Face	933,257
Total	***12,954,801***

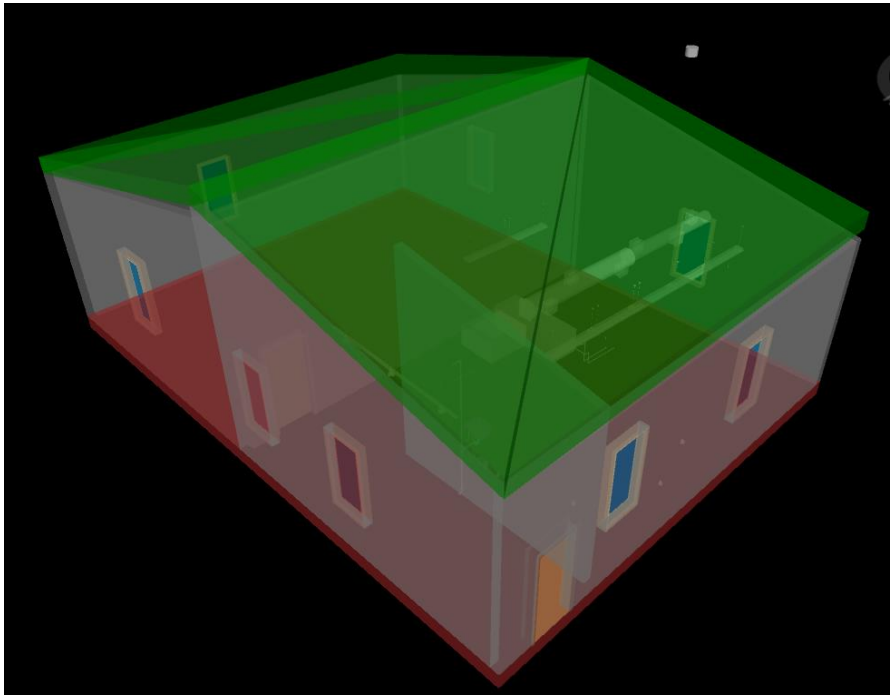
Model-C:

Seven storey hospital model

Architecture	1,385,283 (78.5 MB)	Table	Count
		Elements	65,211
HVAC	1,200,081 (77.2 MB)	Types	2,540
		Properties	390,184
Structural	100,049 (5.6 MB)	Relationships	147,671
Electrical	67,968 (4.7 MB)	Others	4,679
Fire Alarm	15,454 (0.8 MB)	Spatial Index (Level = 11)	26,269,140
Plumbing	375,186 (25.8 MB)		
Sprinkler	502,243 (41.8 MB)	Topology Face	9,443,242
Total	***3,646,264 (234.4 MB)***	***Total***	***36,322,667***

Model-D:

Simple house test model

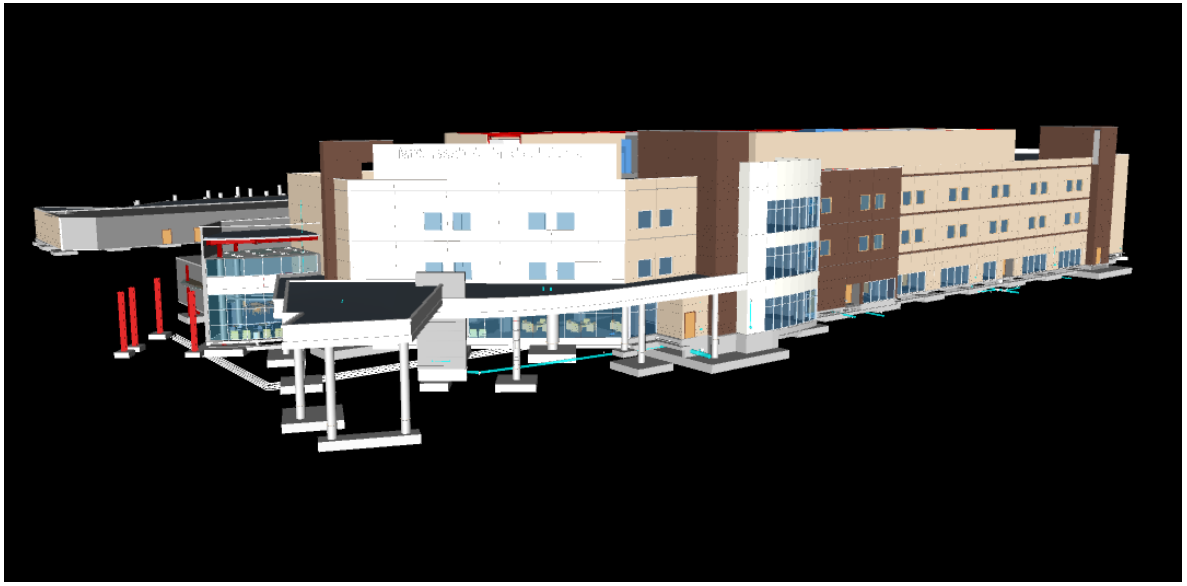


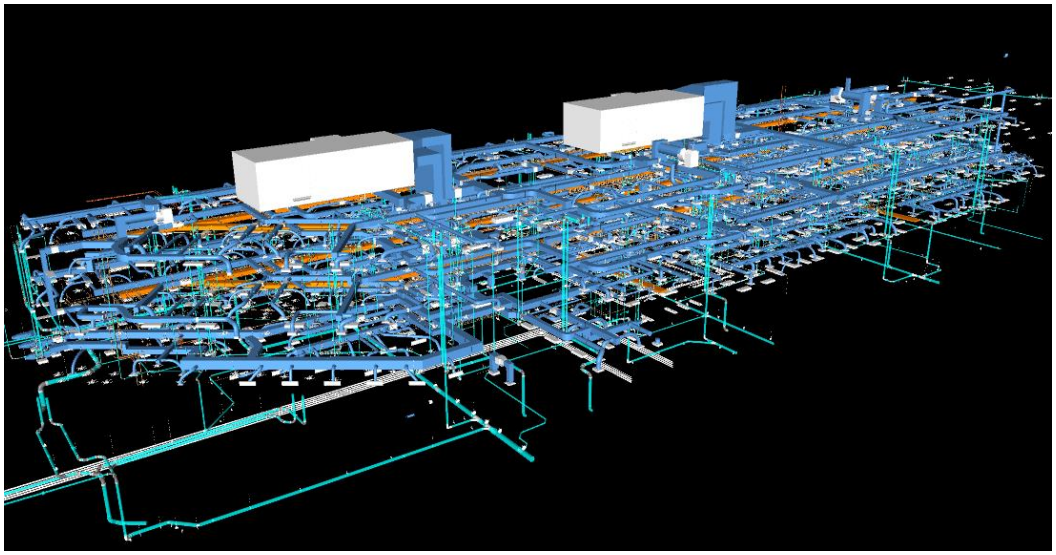
Combined	26,293 (1.3 MB)

Table	Count
Elements	137
Types	42
Properties	243
Relationships	307
Others	624
Total	***1,353***

Model-E

Three Storey Hospital Model



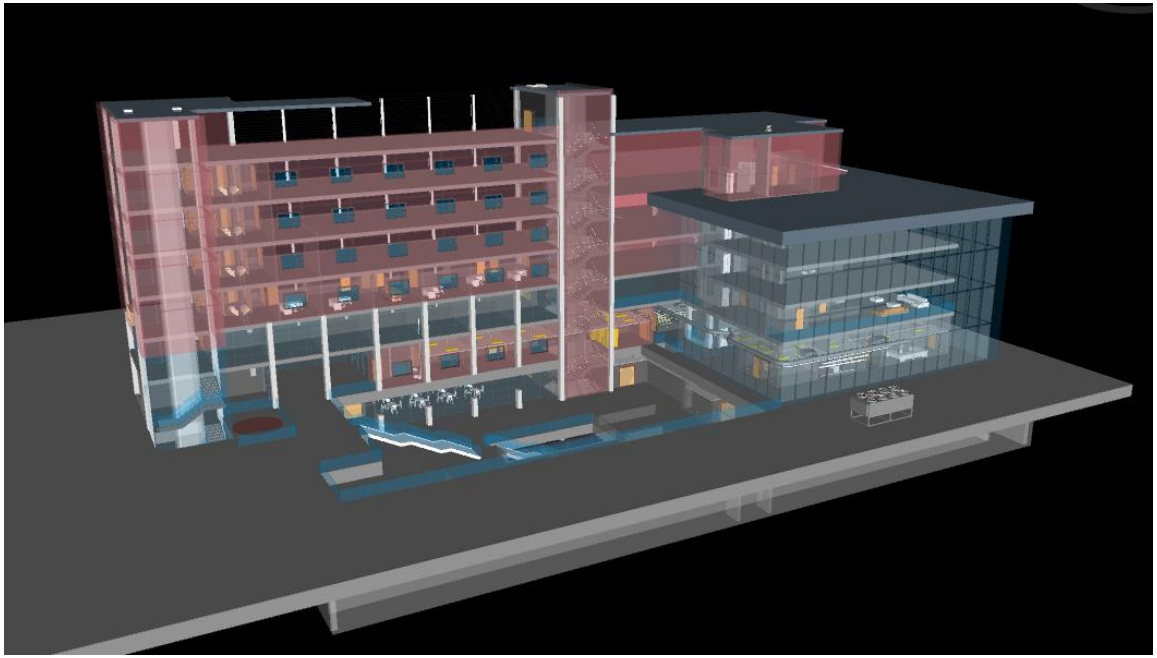


Architecture	651,789 (40.0 MB)	
Electrical	186,821 (11.0 MB)	
HVAC	4,474,564 (253.0 MB)	
Medical_Gas	81,588 (6.2 MB)	
Plumbing (water)	2,225,940 (120.6 MB)	
Plumbing (W&V)	1,211,313 (69.9 MB)	
Structure	206,296 (12.2 MB)	
Total	***9,038,311 (512.9 MB)***	

Table	Count
Elements	45,407
Types	5,136
Properties	335,018
Relationships	92,095
Others	16,180
Spatial Index (Level = 10)	13,563,561
Topology Face	7,185,421
Total	***21,242,818***

Model-F

Seven-Storey Conference Centre



Architecture	1,268,139 (68.7 MB)	
Total	***1,268,139 (68.7 MB)***	

Table	Count
Elements	7,032
Types	790
Properties	109,899
Relationships	19,953
Others	3,265
Spatial Index (Level = 10)	29,418,685
Topology Face	860,091
Total	***30,419,715***

APPENDIX C

BIMRL GRAMMAR IN BNF FORMAT

```
grammar bimrl;

 @lexer::members
 {
       public static int WHITESPACE = 1;
       public static int COMMENTS = 2;
 }

/*
 * Parser Rules
 */
test_rule:    TEST ( assignment_stmt
                   | show_stmt
                   | reset_all_var
                   | expr
                   | color_spec
                   | function
              )
              | start_rule
              | test_rule ';' test_rule ';'
              ;

start_rule:   bimrl_rule ( bimrl_rule )*;

bimrl_rule:   assignment_stmt ';'
              | show_stmt ';'
              | reset_all_var ';'
              | (bimrl_triplets)+
              | sql_stmt ';'
              | delete_var ';'
              | delete_model ';'
              ;

bimrl_triplets:      check_stmt evaluate_stmt? action_stmt? ;

show_stmt:           SHOW VARS
                     ;

reset_all_var:       RESET VARS
                     ;

delete_var:          DELETE VARNAME
                     ;

assignment_stmt:     DEFINE varname assign value
                     | DEFINE varnamelist select_expr
                     | SETVAR varname assign value
                     ;

assign:              ':=';
```

```
select_expr:          select_stmt
                      | '(' select_expr ')'
                      | select_expr select_boolean select_expr
                      ;

select_stmt:          SELECT ( UNIQUE | DISTINCT | ALL )? column_list FROM
                             table_list ( where_clause )? ;

select_boolean:       UNION (ALL)?
                      | INTERSECT
                      | MINUS
                      ;

where_clause:         WHERE expr ;

group_clause:         GROUP BY simple_id_list
                      | ORDER BY simple_id_list (ASCENDING | DESCENDING)?
                      ;

sql_stmt:             SQLANYTHING ;  // For SQL statement, we will just pass the
statement through without changing anything, except renaming the table name with
hex id suffix

varname:              VARNAME ;

varnamelist:          varname ( ',' varname )*
                      | '(' varnamelist ')'
                      ;

delete_model:         DELETE MODEL '(' ( FEDID '=' INT | project_name_number ','
                             project_name_number) ')' ;

project_name_number: PROJECTNAME '=' stringliteral
                      | PROJECTNUMBER '=' stringliteral
                      ;

check_stmt:           CHECK single_check_stmt ';'
                      | CHECK ( multi_check_stmt ';' )+
                      ;

single_check_stmt:    id_list ( where_clause )? (collect_stmt)? ;

collect_stmt:         COLLECT id_list (group_clause)*;

multi_check_stmt:     '{' single_check_stmt '}' AS setname ;

evaluate_stmt:        EVALUATE expr ( output )? (FOREACH GROUP OF (AGGREGATE)?
                             '(' simple_id_list ')' )? (FROM set_clause)?
                             (join_clause)? construct? ';'
                      | EVALUATE ( foreach_fnexpr )+
                      ;

foreach_fnexpr:       '{' expr output (FOREACH GROUP OF (AGGREGATE)?
                             '(' simple_id_list ')' )? (FROM set_clause)?
                             (join_clause)? construct? '}' ';' ;

output:               OUTPUT varname;
```

```
join_clause:        ( LEFT | RIGHT | FULL )? ( OUTER )? JOIN set_clause ( ON
                          '(' id_dot '=' id_dot (AND id_dot '=' id_dot)* ')'
                          | USING '(' id (',' id)* ')' );

set_clause:         setname
                    | '(' select_stmt ')'
                    ;

action_stmt:        ACTION (( one_action ';') | ( multi_action ';' )+) ;

multi_action:       when_clause '{' one_action '}' ;

one_action:         print_action (draw_action)?
                    | draw_action (print_action)?
                    ;

print_action:       PRINT (RESULT | simple_id_list)? (save_action)?;

draw_action:        DRAW (RESULT)? (COLOR color_spec)? (background_model)?
                          (highlight_object)? save_action ;

highlight_object:   HIGHLIGHT '(' (simple_id_list | value_list) ')'
                          (COLOR color_spec)? ;

background_model:   WITH ( transparent )? BACKGROUND id_list (where_clause)? ;

transparent:        (NUMBER)? TRANSPARENT ;

save_action:        SAVE INTO save_destination ( AND save_destination )? ;

save_destination:   X3D x3dfilename
                    | TABLE table_name (APPEND)?
                    ;

bcffilename:        stringliteral;
x3dfilename:        stringliteral;

value_list:         value ( ',' value )* ;

construct:          CONSTRUCT alias '(' geometry_type ')' ;

geometry_type:      LINE '(' ( alias | three_position) (','
                          (alias | three_position))+ ')'
                    | LINE '(' alias (offset)? ',' alias (offset)? ')'
                    | BOX '(' three_position three_position ')'
                    | EXTRUSION '(' face_spec ',' direction ',' extrusion ')'
                    | BREP '(' VERTICES '(' three_position (',' three_position)*
                          ')' ',' face_indexes (',' noVertInFace)? ')'
                    | BREP '(' STARTENDFACES '(' face_spec ',' face_spec ')' ')'
                    | BREP '(' FACESET '(' (face_spec)+ ')' ')'
                    | BREPFROMEDGE '(' face_spec ',' depth ',' extrusion
                          (',' segmentize)? ')'
                                ;

direction:          sign? (XAXIS | YAXIS | ZAXIS | normal |
                          VECTOR three_position )     ;

three_position:     '(' signed_number ',' signed_number ',' signed_number ')' ;
```

```
face_indexes:        FACEINDEXES '(' INT (',' INT)* ')' ;

face_spec:           DEFFACE '(' (alias | (three_position (','
                            three_position)+)) ')' offset? (extend)?
                     | DEFPOINT ('(' alias ')' | three_position ) offset?
                     ;

depth:               signed_number;

extrusion:           signed_number | arithmetic_expr ;

arithmetic_expr:     alias arithmetic_ops alias;

extend:              EXTEND signed_number ( XEDGE | YEDGE | BOTHDIRECTION )?;

segmentize:          NUMBER ;

offset:              OFFSET (three_position | ('(' alias ',' signed_number ')') |
                            ('(' normal ',' signed_number ')') ) ;

normal:              NORMAL '(' alias ')';

noVertInFace:        INTEGER;

when_clause:         WHEN expr;

table_list:          id_list ;

column_list:         id_list | all_columns ;

all_columns:         '*' ;

table_name:          id ;

function_name:       id ;

type_name:           id ;

setname:             id ;

property:            id_dot ;

alias:               id ;

ext_id_dot_notation: id ('.' id)*
                     | (id '.')? function ('.' property)?
                     ;

id_dot:              id ('.' id)* ;

simple_id_list:      id_dot (',' id_dot)* ;

function:            function_name '(' ( (UNIQUE)? expr ( ',' expr )*
                                       | '*' | ) ')' ;

id_list:             id_member (',' id_member)* ;
```

```
id_array:           '(' id (',' id )* ')' ;

id_member:          (id_array | stringliteral | ext_id_dot_notation) (alias)? ;

id:                 STRINGDOUBLEQUOTE
                    | ID
                    ;

pattern:            STRING ;

value:              realliteral
                    | stringliteral
                    | BOOLEAN
                    | NULL;

stringliteral:      STRING ;

realliteral:        signed_number ;

color_spec:         ( RGB '(' NUMBER ',' NUMBER ',' NUMBER ')'
                    | RED
                    | GREEN
                    | BLUE
                    | CYAN
                    | MAGENTA
                    | YELLOW
                    | WHITE
                    | BLACK ) (transparency)?
                    ;

transparency:       TRANSPARENCY NUMBER;

expr:               value
                    | ext_id_dot_notation
                    | VARNAME
                    | BINDNAME
                    | unary_operator expr
                    | expr ops expr
                    | '(' expr ')'
                    | varname_with_bind
                    | expr conditional_expr
                    | exists
                    ;

ops:                arithmetic_ops
                    | comparison_ops
                    | logical_ops
                    ;

arithmetic_ops:     MULTIPLY | DIVIDE | ADDITION | SUBTRACT ;

comparison_ops:     LT | LE | GT | GE | EQ | EQ_DBL | NOTEQ | NOTEQ2 | NOT? LIKE
                    | NOT? REGEXP_LIKE ;

logical_ops:        AND | OR ;

unary_operator:     '-'
                    | '+'
```

```
                    | NOT
                    ;

varname_with_bind:  VARNAME BIND ( expr | '(' expr ( ',' expr )* ')' ) ;

conditional_expr:   null_condition
                    | between_condition
                    | in_condition
                    ;

null_condition:     IS NOT? NULL ;

between_condition:  NOT? BETWEEN expr AND expr ;

in_condition:       NOT? IN '(' ( select_expr | expr ( ',' expr )* ) ')' ;

exists:             NOT? EXISTS '(' select_expr ')' ;

/*
 * Lexer rules
*/

// Command tokens

TEST:               [Tt][Ee][Ss][Tt] ;

ACTION:             [Aa][Cc][Tt][Ii][Oo][Nn] ;
APPEND:             [Aa][Pp][Pp][Ee][Nn][Dd] ;
CHECK:              [Cc][Hh][Ee][Cc][Kk] ;
COLLECT:            [Cc][Oo][Ll][Ll][Ee][Cc][Tt] ;
CONSTRUCT:          [Cc][Oo][Nn][Ss][Tt][Rr][Uu][Cc][Tt] ;
CREATE:             [Cc][Rr][Ee][Aa][Tt][Ee] ;
DELETE:             [Dd][Ee][Ll][Ee][Tt][Ee] ;
DEFINE:             [Dd][Ee][Ff][Ii][Nn][Ee] ;
DRAW:               [Dd][Rr][Aa][Ww] ;
EVALUATE:           [Ee][Vv][Aa][Ll][Uu][Aa][Tt][Ee] ;
FROM:               [Ff][Rr][Oo][Mm] ;
HIGHLIGHT:          [Hh][Ii][Gg][Hh][Ll][Ii][Gg][Hh][Tt] ;
INSERT:             [Ii][Nn][Ss][Ee][Rr][Tt] ;
PRINT:              [Pp][Rr][Ii][Nn][Tt] ;
RESET:              [Rr][Ee][Ss][Ee][Tt] ;
SAVE:               [Ss][Aa][Vv][Ee] ;
SELECT:             [Ss][Ee][Ll][Ee][Cc][Tt] ;
SETVAR:             [Ss][Ee][Tt][Vv][Aa][Rr] ;
SHOW:               [Ss][Hh][Oo][Ww] ;
SQL:                [Ss][Qq][Ll] ;
UPDATE:             [Uu][Pp][Dd][Aa][Tt][Ee] ;
WHEN:               [Ww][Hh][Ee][Nn] ;
WHERE:              [Ww][Hh][Ee][Rr][Ee] ;

// Keywords:
AGGREGATE:          [Aa][Gg][Gg][Rr][Ee][Gg][Aa][Tt][Ee] ;
ALIGN:              [Aa][Ll][Ii][Gg][Nn] ;
ALL:                [Aa][Ll][Ll] ;
AT:                 [Aa][Tt] ;
AS:                 [Aa][Ss] ;
ASCENDING:          [Aa][Ss][Cc]([Ee][Nn][Dd][Ii][Nn][Gg])? ;
BACKGROUND:         [Bb][Aa][Cc][Kk][Gg][Rr][Oo][Uu][Nn][Dd] ;
```

```
BCFFILE:            [Bb][Cc][Ff][Ff][Ii][Ll][Ee] ;
BETWEEN:            [Bb][Ee][Tt][Ww][Ee][Ee][Nn] ;
BIND:               [Bb][Ii][Nn][Dd] ;
BLACK:              [Bb][Ll][Aa][Cc][Kk] ;
BLUE:               [Bb][Ll][Uu][Ee] ;
BOOLEANTYPE:        [Bb][Oo][Oo][Ll][Ee][Aa][Nn] ;
BREP:               [Bb][Rr][Ee][Pp] ;
BOTHDIRECTION:      [Bb][Oo][Tt][Hh][Dd][Ii][Rr][Ee][Cc][Tt][Ii][Oo][Nn] ;
BOX:                [Bb][Oo][Xx] ;
BY:                 [Bb][Yy] ;
CYAN:               [Cc][Yy][Aa][Nn] ;
COLOR:              [Cc][Oo][Ll][Oo][Rr] ;
DEFFACE:            [Dd][Ee][Ff][Ff][Aa][Cc][Ee] ;
DEFFACEFROMEDGE:    [Dd][Ee][Ff][Ff][Aa][Cc][Ee][Ff][Rr][Oo][Mm][Ee][Dd][Gg][Ee] ;
DEFPOINT:           [Dd][Ee][Ff][Pp][Oo][Ii][Nn][Tt] ;
DESCENDING:         [Dd][Ee][Ss][Cc]([Ee][Nn][Dd][Ii][Nn][Gg])? ;
DISTINCT:           [Dd][Ii][Ss][Tt][Ii][Nn][Cc][Tt] ;
DOUBLETYPE:         [Dd][Oo][Uu][Bb][Ll][Ee] ;
ELEMENTSET:         [Ee][Ll][Ee][Mm][Ee][Nn][Tt][Ss][Ee][Tt] ;
EXISTS:             [Ee][Xx][Ii][Ss][Tt][Ss] ;
EXTEND:             [Ee][Xx][Tt][Ee][Nn][Dd] ;
EXTRUSION:          [Ee][Xx][Tt][Rr][Uu][Ss][Ii][Oo][Nn] ;
FACEINDEXES:        [Ff][Aa][Cc][Ee][Ii][Nn][Dd][Ee][Xx][Ee][Ss] ;
FACESET:            [Ff][Aa][Cc][Ee][Ss][Ee][Tt] ;
FEDID:              [Ff][Ee][Dd][Ii][Dd] ;
FOREACH:            [Ff][Oo][Rr][Ee][Aa][Cc][Hh] ;
FULL:               [Ff][Uu][Ll][Ll] ;
GEOMETRY:           [Gg][Ee][Oo][Mm][Ee][Tt][Rr][Yy] ;
GREEN:              [Gg][Rr][Ee][Ee][Nn] ;
GROUP:              [Gg][Rr][Oo][Uu][Pp] ;
IN:                 [Ii][Nn] ;
INTERSECT:          [Ii][Nn][Tt][Ee][Rr][Ss][Ee][Cc][Tt] ;
INTEGERTYPE:        [Ii][Nn][Tt][Ee][Gg][Ee][Rr] ;
INTO:               [Ii][Nn][Tt][Oo] ;
IS:                 [Ii][Ss] ;
JOIN:               [Jj][Oo][Ii][Nn] ;
LEFT:               [Ll][Ee][Ff][Tt] ;
LINE:               [Ll][Ii][Nn][Ee] ;
MAGENTA:            [Mm][Aa][Gg][Ee][Nn][Tt][Aa] ;
MINUS:              [Mm][Ii][Nn][Uu][Ss] ;
MODEL:              [Mm][Oo][Dd][Ee][Ll] ;
NORMAL:             [Nn][Oo][Rr][Mm][Aa][Ll] ;
NOT:                [Nn][Oo][Tt] ;
NULL:               [Nn][Uu][Ll][Ll] ;
OF:                 [Oo][Ff] ;
OFFSET:             [Oo][Ff][Ss][Ee][Tt] ;
ON:                 [Oo][Nn] ;
ORDER:              [Oo][Rr][Dd][Ee][Rr] ;
OUTER:              [Oo][Uu][Tt][Ee][Rr] ;
OUTPUT:             [Oo][Uu][Tt][Pp][Uu][Tt] ;
PLACE:              [Pp][Ll][Aa][Cc][Ee] ;
PROJECTNAME:        [Pp][Rr][Oo][Jj][Ee][Cc][Tt][Nn][Aa][Mm][Ee] ;
PROJECTNUMBER:      [Pp][Rr][Oo][Jj][Ee][Cc][Tt][Nn][Uu][Mm][Bb][Ee][Rr] ;
RED:                [Rr][Ee][Dd] ;
RESULT:             [Rr][Ee][Ss][Uu][Ll][Tt] ;
RIGHT:              [Rr][Ii][Gg][Hh][Tt] ;
RGB:                [Rr][Gg][Bb] ;
START:              [Ss][Tt][Aa][Rr][Tt] ;
```

```
STARTENDFACES:      [Ss][Tt][Aa][Rr][Tt][Ee][Nn][Dd][Ff][Aa][Cc][Ee][Ss] ;
STRINGTYPE:         [Ss][Tt][Rr][Ii][Nn][Gg] ;
TABLE:              [Tt][Aa][Bb][Ll][Ee] ;
TO:                 [Tt][Oo] ;
TRANSPARENCY:       [Tt][Rr][Aa][Nn][Ss][Pp][Aa][Rr][Ee][Nn][Cc][Yy] ;
TRANSPARENT:        [Tt][Rr][Aa][Nn][Ss][Pp][Aa][Rr][Ee][Nn][Tt] ;
UNION:              [Uu][Nn][Ii][Oo][Nn] ;
UNIQUE:             [Uu][Nn][Ii][Qq][Uu][Ee] ;
USING:              [Uu][Ss][Ii][Nn][Gg] ;
VALUES:             [Vv][Aa][Ll][Uu][Ee][Ss] ;
VARS:               [Vv][Aa][Rr][Ss] ;
VECTOR:             [Vv][Ee][Cc][Tt][Oo][Rr] ;
VERTICES:           [Vv][Ee][Rr][Tt][Ii][Cc][Ee][Ss] ;
WHITE:              [Ww][Hh][Ii][Tt][Ee] ;
WITH:               [Ww][Ii][Tt][Hh] ;
X3D:                [Xx][3][Dd] ;
XAXIS:              [Xx][Aa][Xx][Ii][Ss] ;
XEDGE:              [Xx][Ee][Dd][Gg][Ee] ;
YAXIS:              [Yy][Aa][Xx][Ii][Ss] ;
YEDGE:              [Yy][Ee][Dd][Gg][Ee] ;
YELLOW:             [Yy][Ee][Ll][Ll][Oo][Ww] ;
ZAXIS:              [Zz][Aa][Xx][Ii][Ss] ;


/* Operators */
MULTIPLY:           '*';
DIVIDE:             '/';
ADDITION:           '+';
SUBTRACT:           '-';
LT:                 '<';
LE:                 '<=';
GT:                 '>';
GE:                 '>=';
EQ:                 '=' ;
EQ_DBL:             '==' ;
NOTEQ:              '!=' ;
NOTEQ2:             '<>' ;
LIKE:               [Ll][Ii][Kk][Ee];
REGEXP_LIKE:        [Rr][Ee][Gg][Ee][Xx][Pp][_][Ll][Ii][Kk][Ee] ;
OR:                 [Oo][Rr] | '||' ;
AND:                [Aa][Nn][Dd] | '&&';

// Variable tokens
VARNAME:            [?] ALPHANUMERIC+ ;
BINDNAME:           [:] ALPHANUMERIC+ ;
ID:                 [a-zA-Z] ALPHANUMERIC* ;  // valid ID only starts with
alphabet
STRING :            ['] (ESC | .)*? ['] ;
STRINGDOUBLEQUOTE: '"' (ESC | .)*? '"' ;

fragment ALPHANUMERIC:      [a-zA-Z0-9_] ;
fragment ESC:       '\\' (["\\/bfnrt] | UNICODE) ;
fragment UNICODE :  'u' HEX HEX HEX HEX ;
fragment HEX :      [0-9a-fA-F] ;

sign:               '+' | '-' ;
signed_number:      ( '+' | '-' )? NUMBER ;
signed_integer:     ( '+' | '-' )? INTEGER ;
```

```
NUMBER:              INT '.' INT? EXP?   // 1.35, 1.35E-9, 0.3
                     | '.' INT EXP?                    // .2, .2e-9
                     | INT EXP?            // 1e10
                     | INT                // 45
                     ;

BOOLEAN :            [.][Tt][Rr][Uu][Ee][.]
                     | [.][Tt][.]
                     | [.][Ff][Aa][Ll][Ss][Ee][.]
                     | [.][Ff][.]
                     ;

INCR:                '++'[0-9]* ;
DECR:                '--'[0-9]* ;
INTEGER:             INT ;
fragment INT:        [0] | [0-9] [0-9]* ;
fragment EXP:        [Ee] [+\-]? INT ;

SQLANYTHING:         [Ss][Qq][Ll] .*? ';' ;

WS:                  [ \t\n\r]+ -> channel(WHITESPACE) ;

SINGLE_LINE_COMMENT: '//' ~[\r\n]* -> channel(COMMENTS) ;

MULTILINE_COMMENT:   '/*' .*? ( '*/' | EOF ) -> channel(COMMENTS) ;
```

REFERENCES

1. Lee, J.K., *Building environment rule and analysis (BERA) language and its application for evaluating building circulation and spatial program*, in *College of Architecture*. 2011, Georgia Institute of Technology.
2. Khemlani, L. *CORENET e-PlanCheck: Singapore's Automated Code Checking System*. http://www.aecbytes.com/buildingthefuture/2005/CORENETePlanCheck.html 2005 [cited 2014 03/01/2014].
3. Harper, R.F., *The Code of Hammurabi King of Babylon*. Second Edition ed. 1904: The University of Chicago Press.
4. Fenves, S.J., *Tabular decision logic for structural design.* Proceedings of the American Society of Civil Engineers, 1966. **92**: p. 473-490.
5. Dimyadi, J. and R. Amor. *Automated Building Code Compliance Checking - Where is it at?* in *CIB WBC 2013*. 2013. Brisbane, Australia.
6. Eastman, C., et al., *Automatic rule-based checking of building designs.* Automation in Construction, 2009. **18**(8): p. 1011-1033.
7. Nawari, N. *The Challenge of Computerizing Building Codes in BIM Environment*. in *Computing in Civil Engineering: Proceedings of the 2012 Asce International Conference on Computing in Civil Engineering*. 2012. Clearwater Beach, Florida; Jun 17-20, 2012: ASCE Publications.
8. Goel S. K., F.S.J., *Computer-aided Processing of Structural Design Specifications.* Structural Design, University of Illinois at Urbana-Champaign, 1969.
9. Hjelseth, E. and N. Nisbet. *Exploring semantic based model checking*. in *Proceedings of the 2010 27th CIB W78 International Conference*. 2010.
10. Hjelseth, E. and N. Nisbet. *Capturing normative constraints by use of the semantic mark-up (RASE) methodology*. in *CIB W78 2011 28th International Conference-Applications of IT in the AEC Industry*. 2011.
11. Zhang, J. and N. El-Gohary, *Extraction of Construction Regulatory Requirements from Textual Documents Using Natural Language Processing Techniques.* Proc., Comput. Civ. Eng, 2012: p. 453-460.
12. Salama, D.A. and N.M. El-Gohary, *Towards automated compliance checking of construction operation plans using a deontology for the construction domain.* Journal of Computing in Civil Engineering, 2013.
13. Salama, D.M. and N.M. El-Gohary, *Semantic Text Classification for Supporting Automated Compliance Checking in Construction.* Journal of Computing in Civil Engineering, 2013.
14. Pauwels, P., et al., *A semantic rule checking environment for building performance checking.* Automation in Construction, 2011. **20**(5): p. 506-518.
15. Solibri. *Solibri Model Checker*. http://www.solibri.com/solibri-model-checker/functionality-highlights.html [cited 2014 03/01/2014].
16. novaCITYNETS. *novaSPRINT awarded CORENET Integrated Plan Checking System*. http://www.novacitynets.com/news.htm 2000 [cited 2014 03/01/2014].

17. novaCITYNETS. *FORNAX Roll-out in Norway*. http://www.novacitynets.com/news_2005nov.htm 2005 [cited 2014 03/01/2014].
18. novaCITYNETS, *FORNAX Makes Headway in Kingdom of Saudi Arabia*, in *http://www.novacitynets.com/news_2013dec.htm*. 2013.
19. Malsane, S., et al., *Development of an object model for automated compliance checking.* Automation in Construction, 2015. **49**: p. 51-58.
20. Ding, L., et al. *Automated code checking*. in *CRC CI International Conference 2004*. 2004.
21. Ding, L., et al., *Automating code checking for building designs-DesignCheck.* 2006.
22. Beach, T., et al., *Towards Automated Compliance Checking in the Construction Industry*, in *Database and Expert Systems Applications*, H. Decker, et al., Editors. 2013, Springer Berlin Heidelberg. p. 366-380.
23. Nawari, N.O. *Automating Codes Conformance in Structural Domain*. in *ASCE International Workshop on Computing in Civil Engineering*. 2011. Miami, Florida, June 19-22, 2011: American Society of Civil Engineers.
24. Park, J. and Y. Lee, *Developing a Performance-Based Design System with Semantic Interoperability*, in *HCI International 2013 - Posters' Extended Abstracts*, C. Stephanidis, Editor. 2013, Springer Berlin Heidelberg. p. 69-73.
25. Choi, J., J. Choi, and I. Kim, *Development of BIM-based evacuation regulation checking system for high-rise and complex buildings.* Automation in Construction, 2014. **46**: p. 38-49.
26. Preidel, C. and A. Borrmann, *Automated Code Compliance Checking Based on a Visual Language and Building Information Modeling*, in *International Symposium on Automation and Robotics in Construction and Mining (ISARC 2015)*. 2015: Oulu, Finnland.
27. Zhang, S., et al., *Building information modeling (BIM) and safety: Automatic safety checking of construction models and schedules.* Automation in Construction, 2013. **29**: p. 183-195.
28. Martins, J.P. and A. Monteiro, *LicA: A BIM based automated code-checking application for water distribution systems.* Automation in Construction, 2013. **29**: p. 12-23.
29. Liu, X., et al., *Domain-Specific Querying Formalisms for Retrieving Information of HVAC Systems.* Journal of Computing in Civil Engineering, 2014. **28**(1): p. 40-49.
30. Tan, X., A. Hammad, and P. Fazio, *Automated code compliance checking for building envelope design.* Journal of Computing in Civil Engineering, 2010. **24**(2): p. 203-211.
31. Lam, K.P., et al., *Mapping of industry building product model for detailed thermal simulation and analysis.* Advances in Engineering Software, 2006. **37**(3): p. 133-145.
32. Lee, J.M., *Automated checking of building requirements on circulation over a range of design phases*, in *College of Architecture*. 2010, Georgia Institute of Technology.
33. Lê, M., et al. *The HITOS project-A full scale IFC test*. in *Proceedings of the 2006 ECPPM Conference*. 2006.

34. Borrmann, A. and E. Rank, *Topological analysis of 3D building models using a spatial query language.* Advanced Engineering Informatics, 2009. **23**(4): p. 370-385.
35. Borrmann, A. and E. Rank, *Query support for BIMs using semantic and spatial conditions.* Handbook of research on building information modeling and construction informatics: Concepts and technologies, 2010: p. 405-450.
36. Borrmann, A., S. Schraufstetter, and E. Rank, *Implementing metric operators of a spatial query language for 3D building models: octree and B-Rep approaches.* Journal of Computing in Civil Engineering, 2009. **23**(1): p. 34-46.
37. Borrmann, A., et al. *An iterative, octree-based algorithm for distance computation between polyhedra with complex surfaces*. in *Proc. of the Int. ASCE Workshop on Computing in Civil Engineering*. 2007.
38. Borrmann, A., et al. *An octree-based implementation of directional operators in a 3D spatial query language for building information models*. in *Proc. 24th W78 Conf. Maribor*. 2007.
39. Borrmann, A., C. Van Treeck, and E. Rank. *Towards a 3D spatial query language for building information models*. in *Proc. Joint Int. Conf. of Computing and Decision Making in Civil and Building Engineering (ICCCBE-XI)*. 2006.
40. Daum, S. and A. Borrmann, *Processing of Topological BIM Queries using Boundary Representation Based Methods.* Advanced Engineering Informatics, 2014.
41. Beetz, J., et al. *BIMserver. org–An open source IFC model server*. in *Proceedings of the CIP W78 conference*. 2010.
42. Mazairac, W. and J. Beetz, *BIMQL–An open query language for building information models.* Advanced Engineering Informatics, 2013.
43. Solihin, W., *Lessons learned from experience of code-checking implementation in Singapore.* avialable online at https://www.researchgate.net/publication/280599027_Lessons_learned_from_experience_of_code-checking_implementation_in_Singapore, 2004.
44. Solihin, W., et al., *Beyond Interoperability of Building Models: A Case for Code Compliance Checking*, in *BP-CAD Workshop*. 2004, available online at https://www.researchgate.net/publication/280598933_BEYOND_INTEROPERATIBILITY_OF_BUILDING_MODEL_A_CASE_FOR_CODE_COMPLIANCE_CHECKING: Carnegie Melon University.
45. Solihin, W. and N.A. Shaikh, *Improving Application Resiliency the Face of Imperfect Information*, in *International Workshop on nD Modelling Road map: A Vision for nD-Enabled Construction*. 2005: University of Salford.
46. Hietanen, J. and S. Final, *IFC model view definition format.* International Alliance for Interoperability, 2006.
47. Solihin, W. and C. Eastman, *A Knowledge Representation Approach to Capturing BIM Based Rule Checking Requirements Using Conceptual Graph*, in *CIB W78 Conference 2015*. 2015: Eindhoven, the Netherland.
48. Kato, K., et al. *Mask rule check using priority information of mask patterns*. in *27th Annual BACUS Symposium on Photomask Technology*. 2007. International Society for Optics and Photonics.
49. Siemens, *Siemens NX 10 Documentation*. 2014.

50. Solihin, W., C. Eastman, and Y.-C. Lee, *Toward robust and quantifiable automated IFC quality validation.* Advanced Engineering Informatics, 2015. **29**(3): p. 739-756.
51. Anjali, J. and R. Mahbub, *The architecture of safety: hospital design* Current Opinion in Critical Care, 2007. **13**: p. 6.
52. Lu, Y., J. Peponis, and C. Zimring. *Targeted Visibility Analysis in Buildings*. in *7th International Space Syntax Symposium*. 2009. Stockholm.
53. Reiling, J., *Safe design of healthcare facilities.* Quality & Safety in Health Care, 2006. **15**(Suppl 1): p. i34-i40.
54. BCA. *Singapore BIM Guide*. http://www.corenet.gov.sg/integrated_submission/bim/BIM/Singapore%20BIM%20Guide_V2.pdf 2013; Version 2:[
55. Hietanen, J. and S. Lehtinen, *The useful minimum.* Tampere University of Technology, Tampere, 2006.
56. BuildingSMART. *IFC Solutions Factory - The Model View Definition site*. http://www.blis-project.org/IAI-MVD/.
57. BIMForum, *Level of Development Specification - for Building Information Models*. 2013, BIMForum: http://bimforum.org/wp-content/uploads/2013/08/2013-LOD-Specification.pdf.
58. Fiatech, *AutoCodes Project: Phase 1, Proof-of-Concept Final Report*, in *http://www.fiatech.org/images/stories/techprojects/project_deliverables/Updated_project_deliverables/AutoCodesPOCFINALREPORT.pdf*. 2012.
59. Fiatech. *Automated Code Plan Checking Tool-Proof-Of-Concept (Phase 2)*. http://www.fiatech.org/project-management/projects/593-automated-code-plan-checking-tool-proof-of-concept#sthash.RaV9fZnR.dpuf [cited 2014 03/01/2014].
60. Nawari, N.O., *Automated Code Checking in BIM Environment*, in *14th International Conference on Computing in Civil and Building Engineering*, http://www.icccbe.ru/paper_long/0279paper_long.pdf, Editor. 2012: Moscow, Russia, 27 - 29 June, 2012.
61. Hjelseth, E., *Foundation for development of computable rules.* Norwegian University of Life Sciences (UMB), Dept. of Mathematical Sciences and Technology, Norway, 2009.
62. Zhong, B., et al., *Ontology-based semantic modeling of regulation constraint for automated construction quality compliance checking.* Automation in Construction, 2012. **28**: p. 58-70.
63. Salama, D. and N. El-Gohary. *Semantic modeling for automated compliance checking*. in *Proc., 2011 ASCE Intl. Workshop on Comput. in Civ. Eng*. 2011.
64. Yurchyshyna, A. and A. Zarli, *An ontology-based approach for formalisation and semantic organisation of conformance requirements in construction.* Automation in Construction, 2009. **18**(8): p. 1084-1098.
65. BuildingSMART, *IFC2x3 CV2.0 Certification sub-schema (MVD)*, in *http://www.buildingsmart-tech.org/specifications/ifc-view-definition/coordination-view-v2.0/sub-schema*.
66. Singapore, B., *IFC2x2 Code Checking Certification View*, in *http://www.ifcwiki.org/index.php/IFC_Certified_Software*. 2005.

67. ICC, *IBC 2009*, in *http://shop.iccsafe.org/codes/2009-international-codes/2009-international-building-code-tab-combo.html*. 2009.
68. Singapore, *Singapore Building Codes*, in *http://www.corenet.gov.sg/einfo/Index.aspx*.
69. BuildingSMART, *FM Basic Handover*, in *http://www.buildingsmart-tech.org/specifications/ifc-view-definition/fm-handover-aquarium/fm-basic-handover*. 2009: http://www.buildingsmart-tech.org/specifications/ifc-view-definition/fm-handover-aquarium/fm-basic-handover.
70. ICC, *ICC IFC 2003 Commentary*, in *http://www.constructionbook.com/2003-icc-international-fire-code-ifc-commentary-3410s03/2003-international/*. 2003.
71. Fortune, S., *Voronoi diagrams and Delaunay triangulations.* Computing in Euclidean geometry, 1992. **1**: p. 193-233.
72. Hu, Z., J. Zhang, and Z. Deng, *Construction Process Simulation and Safety Analysis Based on Building Information Model and 4D Technology.* Tsinghua Science & Technology, 2008. **13, Supplement 1**(0): p. 266-272.
73. Behm, M., *Linking construction fatalities to the design for construction safety concept.* Safety Science, 2005. **43**(8): p. 589-611.
74. Dewlaney, K.S., M.R. Hallowell, and B.R. Fortunato III, *Safety risk quantification for high performance sustainable building construction.* Journal of Construction Engineering and Management, 2011. **138**(8): p. 964-971.
75. Delis, E.A. and A. Delis, *Automatic fire-code checking using expert-system technology.* Journal of computing in civil engineering, 1995. **9**(2): p. 141-156.
76. Singapore, *Singapore Fire Code 2013*, in *http://www.scdf.gov.sg/content/scdf_internet/en/building-professionals/publications_and_circulars/fire-code-2013.html*. 2013.
77. GSA, *Courts Design Guide*, in *http://www.gsa.gov/graphics/pbs/Courts_Design_Guide_07.pdf*.
78. OSHA, *Fall Protection in Construction.* 1998. **OSHA 3146**.
79. buildingSMART. *Global Testing Documentation Server (GTDS)*. http://gtds.buildingsmart.com [cited 2014 03/01/2014].
80. Alchemy, D. *IFC BIM Validation Service*. http://www.digitalalchemypro.com/html/services/IfcBimValidationService.html [cited 2014 03/01/2014].
81. Autodesk. *Navisworks clash detection*. http://www.autodesk.com/products/autodesk-navisworks-family/features [cited 2014 03/01/2014].
82. Tekla. *Tekla BIMsight clash detection feature*. http://www.teklabimsight.com/helpcenter/designCoordinationTopic.jsp?topic=dcg_conflict_checks [cited 2014 03/01/2014].
83. Autodesk. *Revit Model Review*. http://wikihelp.autodesk.com/Revit/enu/2013/Help/00005-More_Inf0/0001-Subscrip1/0035-Autodesk35 [cited 2014 03/01/2014].
84. Autodesk. *Navisworks search set feature*. http://www.augi.com/library/power-up-with-search-and-selection-sets [cited 2014 03/01/2014].
85. AEC3. *Xabio*. http://www.aec3.com/en/5/5_010_XABIO.htm [cited 2014 03/01/2014].

86. Fiatech, *Fiatech Autocode phase 1 report.* 2012.
87. ISO, *ISO 16739:2013 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries*. 2013: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=51622.
88. Sowa, J.F., *Conceptual graphs for a data base interface.* IBM Journal of Research and Development, 1976. **20**(4): p. 336-357.
89. Sowa, J.F., *Conceptual Structures - Information Processing in Mind and Machine*. 1984: Addison-Wesley Publishing Company.
90. Chein, M. and M.-L. Mugnier, *Graph-based knowledge representation: Computational foundations of conceptual graphs*. 2008: Springer.
91. Solihin, W. and C. Eastman, *Classification of rules for automated BIM rule checking development.* Automation in Construction, 2015. **53**(0): p. 69-82.
92. Malsane, S., et al., *Development of an object model for automated compliance checking.* Automation in Construction, 2015. **49, Part A**(0): p. 51-58.
93. Forgy, C.L., *On the efficient implementation of production systems*. 1979, Carnegie-Mellon University.
94. Forgy, C.L., *Rete: A fast algorithm for the many pattern/many object pattern match problem.* Artificial Intelligence, 1982. **19**(1): p. 17-37.
95. Loffredo, D. and M. Hardwick, *Efficient database implementation of EXPRESS information models.* May, 1998. **1**: p. 1-133.
96. BLIS. *BLIS (Building Lifecycle Interoperable Software)*. http://blis-project.org/index2.html 2002.
97. Adachi, Y., *Technical Overview of IFC Model Server*, in *online at http://cic.vtt.fi/projects/ifcsvr/tec/VTT-TEC-ADA-11.pdf*. 2002.
98. You, S.-J., D. Yang, and C.M. Eastman. *Relational DB Implementation of STEP based product model*. in *CIB World Building Congress 2004*. 2004.
99. Kang, H.-s. and G. Lee, *Development of an object-relational ifc server.* ICCEM/ICCPM, 2009.
100. EPM. *EDM Model Server (IFC)*. http://www.epmtech.jotne.com/products/edm-model-server-ifc 2014 [cited 2014.
101. Lee, G., et al., *Query Performance of the IFC Model Server Using an Object-Relational Database (ORDB) Approach and a Traditional Relation Database (RDM) Approach.* Journal of Computing in Civil Engineering, 2012.
102. Jeong, J., G. Lee, and H. Kang. *Preliminary Performance Evaluation of an ORDB-based IFC server and an RDB-based IFC server by using the BUCKY Benchmark Method*. in *Proceedings of CIB World Congress*. 2010.
103. Adamson, C., *Star schema*. 2010: McGraw-Hill.
104. Kimball, R. and M. Ross, *The data warehouse toolkit: the complete guide to dimensional modeling*. 2011: John Wiley & Sons.
105. Bonifati, A., et al., *Designing data marts for data warehouses.* ACM transactions on software engineering and methodology, 2001. **10**(4): p. 452-483.
106. Levene, M. and G. Loizou, *Why is the snowflake schema a good data warehouse design?* Information Systems, 2003. **28**(3): p. 225-240.
107. Shin, S.K. and G.L. Sanders, *Denormalization strategies for data retrieval from data warehouses.* Decision Support Systems, 2006. **42**(1): p. 267-282.

108. C, E., *An interrogation language for building descriptions.* LUBFS Coference Processdings Number 2, 1975.
109. Borrmann, A. and E. Rank, *Specification and implementation of directional operators in a 3D spatial query language for building information models.* Advanced Engineering Informatics, 2009. **23**(1): p. 32-44.
110. Borrmann, A. and J. Beetz. *Towards spatial reasoning on building information models*. in *Proceedings of the 8th European Conference on Product and Process Modeling (ECPPM), Taylor & Francis Group, Cork, Ireland*. 2010.
111. OGC, *OpenGIS® Implementation Standard for Geographic information - Simple feature access*, in *Part 2: SQL option*. 2010: http://portal.opengeospatial.org/files/?artifact_id=25354.
112. Zlatanova, S., *3D geometries in spatial DBMS*, in *Innovations in 3D geo information systems*. 2006, Springer. p. 1-14.
113. PostGIS *PostGIS 2.1.3dev Manual*. http://www.postgis.net/stuff/postgis-2.1.pdf, 2014.
114. Oracle, *Oracle Spatial and Graph Developer's Guide*, in *online at https://docs.oracle.com/database/121/SPATL/toc.htm*. 2014: https://docs.oracle.com/database/121/SPATL/toc.htm.
115. Solihin, W. and C. Eastman, *A Framework for Fully Integrated Building Information Models in a Federated Environment.* Submitted for publication to Advanced Engineering Informatics, 2014.
116. Vebree, E. and S. Zlatanova, *3D-modeling with respect to boundary representations within geo-DBMS*. 2004, GISt report.
117. Breunig, M. and S. Zlatanova, *3D geo-database research: Retrospective and future directions.* Computers and Geosciences, 2011. **37**(7): p. 791-803.
118. Oracle, *Oracle Spatial and Graph Topology Data Model and Network Data Model Graph Developer's Guide*, in *https://docs.oracle.com/database/121/SPAJV/toc.htm*. 2014, Oracle: online at https://docs.oracle.com/database/121/SPAJV/toc.htm.
119. Karnatak, H.C. and V. Kumar, *Performance study of various spatial indexes on 3D geo-data in Geo-RDBMS.* Geocarto International, 2015. **30**(6): p. 633-649.
120. Requicha, A., *Mathematical models of rigid solid objects.* 1977.
121. Requicha, A. and R. Tilove, *Mathematical foundations of constructive solid geometry: general topology of regular closed sets*, in *Tech. Memo. No. 27*. 1978, Production Automation Project, Univ. of Rochester.
122. Requicha, A.G., *Representations for Rigid Solids: Theory, Methods, and Systems.* ACM Comput. Surv., 1980. **12**(4): p. 437-464.
123. Requicha, A.G., *Representations of rigid solid objects*, in *Computer Aided Design Modelling, Systems Engineering, CAD-Systems*, J. Encarnacao, Editor. 1980, Springer Berlin Heidelberg. p. 1-78.
124. Baer, A., C. Eastman, and M. Henrion, *Geometric modelling: a survey.* Computer-Aided Design, 1979. **11**(5): p. 253-272.
125. Eastman, C.M. and K.J. Weiler, *Geometric modeling using the Euler operators.* 1979.
126. Reddy, D.R. and S. Rubin, *Representation of Three-Dimensional Objects*. 1978, DTIC Document.

127. BuildingSMART, *IFC4 Reference View*, in *https://github.com/BuildingSMART/IFC4-CV/blob/master/IFC4_Reference_View.md*. 2014.
128. Guttman, A., *R-trees: a dynamic index structure for spatial searching*. Vol. 14. 1984: ACM.
129. Gaede, V., et al., *Multidimensional access methods.* ACM Comput. Surv., 1998. **30**(2): p. 170-231.
130. Samet, H., *Foundations of multidimensional and metric data structures*. 2006: Morgan Kaufmann.
131. Morton, G.M., *A computer oriented geodetic data base and a new technique in file sequencing*. 1966: International Business Machines Company.
132. Meagher, D., *Geometric modeling using octree encoding.* Computer Graphics and Image Processing, 1982. **19**(2): p. 129-147.
133. Meagher, D.J., *The Octree Encoding Method for Efficient Solid Modeling*. 1982, DTIC Document.
134. Egenhofer, M.J., *A formal definition of binary topological relationships*, in *Foundations of data organization and algorithms*. 1989, Springer. p. 457-472.
135. Egenhofer, M.J., *Deriving the Composition of Binary Topological Relations.* Journal of Visual Languages & Computing, 1994. **5**(2): p. 133-149.
136. Frank, A.U., *Qualitative spatial reasoning about distances and directions in geographic space.* Journal of Visual Languages & Computing, 1992. **3**(4): p. 343-371.
137. Goyal, R. and M.J. Egenhofer, *The direction-relation matrix: A representation for directions relations between extended spatial objects.* the annual assembly and the summer retreat of University Consortium for Geographic Information Systems Science, 1997. **15**: p. 22-81.
138. Codd, E.F., *A relational model of data for large shared data banks.* Commun. ACM, 1970. **13**(6): p. 377-387.
139. Gao, H., H. Liu, and M. Yu, *De-Normalize IFC Model for Data Extraction*, in *AEI 2013*. p. 468-476.
140. W3C, *SPARQL 1.1 Query Language*, in *online at http://www.w3.org/TR/sparql11-query/*. 2013, W3C: online at http://www.w3.org/TR/sparql11-query/.
141. Kashlev, A. and A. Chebotko. *SPARQL-to-SQL Query Translation: Bottom-Up or Top-Down?* in *Services Computing (SCC), 2011 IEEE International Conference on*. 2011. IEEE.
142. Beetz, J., Leeuwen, JP., Vries, B., *Towards a topological reasoning service for IFC-based building information models in a Semantic Web context*. 2006, s.n. 2006.
143. Adachi, Y., *Overview of partial model query language.* Concurrent Engineering: Enhanced Interoperable Systems, 2003: p. 549-555.
144. Nentwich, C. and R. James. *NRL: The Natural Rule Language*. http://nrl.sourceforge.net/ 2011.
145. W3C, *SWRL: A Semantic Web Rule Language - Combining OWL and RuleML*. 2004, W3C: online at http://www.w3.org/Submission/SWRL/.

146. LegalRuleML_TC *LegalRuleML*. Online at https://lists.oasis-open.org/archives/legalruleml/201208/msg00040/LegalRuleML-palmirani2012_-RuleML2012v3.pdf, 2012.
147. Athan, T. *LegalRuleML: from Metamodel to Use Cases*. online at http://www.nicta.com.au/pub?doc=7107.
148. Koonce, D., L. Huang, and R. Judd, *EQL an express query language.* Computers & industrial engineering, 1998. **35**(1): p. 271-274.
149. Huang, L., *EXPRESS Query Language and Templates and Rules: Two languages for advanced Software System Integrations*, in *College of Engineering and Technology*. 1999, Ohio University.
150. Parr, T., *The Definitive ANTLR 4 Reference*. http://www.antlr.org/. 2012: The Pragmatic Bookshelf.
151. Fishking, T., *Integrating BIM and Prefabrication in Healthcare Facility Design*, in *27th Annual AHCA Seminar and 49th Annual FHEA Meeting and Tradeshow*. 2011: Orlando, Florida. p. http://www.ahcaseminar.com/Speaker%20pdfs/FacilityEngineeringSessions/Fishking_Integrating%20BIM%20and%20Prefabrication%20Presentation.pdf.
152. Chiang, Y., *Design Dilemma - Nurse's Stations*. 2010, http://iwsp.human.cornell.edu/files/2013/09/Nurses-stations-qr3u4x.pdf.
153. Leong, P. and J. Kang, *Water contamination: 115 affected*, in *The Straits Times*. 2000: Singapore.

VITA

WAWAN SOLIHIN

Mr. Wawan Solihin has an interesting mixture of academic and industry experiences that contribute to the needed expertise to complete this research. His first degree from Bandung Institute of Technology and his subsequent Master's degree from the National University in Singapore in the Chemical Engineering department do not really give relevant knowledge besides the needed training to be a researcher and foundational exposures to CAD and programming language in the early 1990s. The real knowledge comes from the work experience when he decided to pursue career in the area that he is passionate about, i.e. computer science.

The career started with Siemens Nixdorf in Singapore developing a turnkey GIS application for Singapore Urban Redevelopment Authority. During this project he focused on the use of geo-database technology for GIS maps using RDBMS. He went on to work with SICAD Geomatics in Munich, Germany to further his involvement in the development of the geo-database software. Upon his return to Singapore in the year 2000, he joined novaCITYNETS where he was in charge to develop CORENET ePlanCheck project. This is a new domain that is full of challenges involving code-checking, IFC, geometry, spatial and graphics. The project is still widely respected as the most serious attempt to automate the code compliance checking on building models in the world until today. He was instrumental to bring the project to a completion in 2005 and developed FORNAX as a code checking platform for novaCITYNETS. He went on to work on the development of IFC support in Autodesk for Autocad Architecture/MEP product line for five years and then led the initiative to develop a rule checking capability in Autodesk Navisworks before leaving Autodesk in 2012. He holds two US patents related to the rule-checking concepts during the two years with the development initiative for the rule-checking capability in Navisworks.

In 2013, he began his Ph.D. study in Georgia Tech College of Architecture under the supervision of Prof. Chuck Eastman who is also known to be the father of BIM. Rule Checking has been one of his focus in his research portfolio. With his advice and

mentorship, Mr. Wawan Solihin, develops the initial idea on the topic of data representation and rule-checking into his final form of his Ph.D. thesis. With Prof. Eastman, he published two journal papers and one international conference paper. This is in addition of several conference papers he published and presented during the course of development of CORENET ePlanCheck. He also co-taught a graduate course with Prof. Eastman on the topic of rule checking within “COA-8690 Building Product Models” course. He also has submitted one more journal paper and has written several others that are prepared for submissions as journal papers. All these are done in a record fast track pace of just a little more than two years including one year mandatory residency and course works in Georgia Tech, Atlanta.